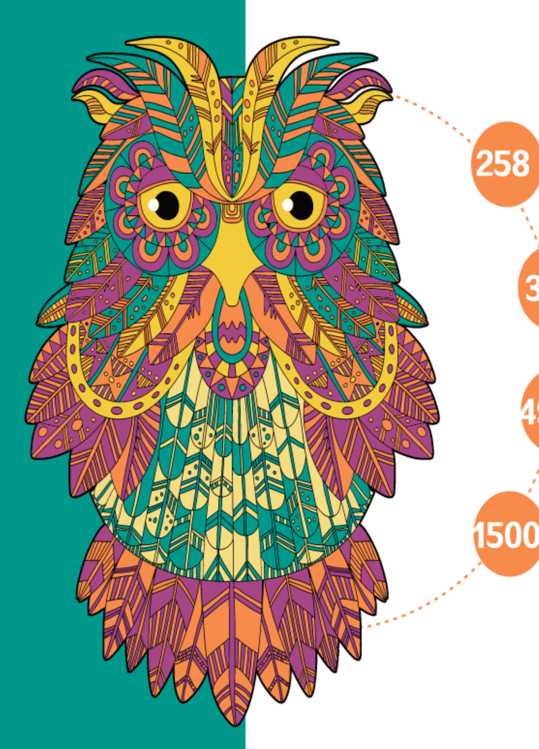




258 节高清微视频 + 6项拓展微阅读 + 310个在线微练习
移动端/PC端同步学习，QQ群/微信群随时答疑。

378 个实例案例分析+249项实例源代码
速查、高效、实用，增强实战能力。

4900 个前端案例 + 48本参考手册
先观摩，再临摹，高手案头常备，随时查阅提升。

1500 套网页模板 + 12000个设计素材 + 1036道前端面试真题
随用随取，提升设计效率，快速进阶开发高手行列。

CSS3 网页设计从入门到精通

微课精编版

前端科技◎编著

线上资源，方便快捷

- 教学微视频（258节）
- 拓展微阅读（6项）
- 在线微练习（310个）
- 权威参考 (21个)

海量资源，可查可用

- 前端案例资源库（4900个）
- 面试题库（1036道）
- 网页模板库（1500套）
- 设计素材库（12000个）

清华大学出版社



CSS3 网页设计从入门到精通（微课精编版）

前端科技　编著

清华大学出版社
北　京

内 容 简 介

《CSS3 网页设计从入门到精通（微课精编版）》从初学者角度出发，通过通俗易懂的语言、丰富多彩的实例，系统讲解了 CSS3 基础理论和实际运用技术。全书共 18 章，包括 CSS3 基础、CSS3 选择器、使用 CSS3 美化文本和图像、使用 CSS3 设计特效文本、使用 CSS 设计背景样式、使用 CSS3 美化列表和超链接样式、使用 CSS 美化表格、使用 CSS 美化表单、CSS 盒模型、使用 CSS 布局网页、CSS3 伸缩盒布局、设计 CSS3 用户界面样式、设计 CSS3 动画、使用 CSS3 媒体查询、使用 JavaScript 控制 CSS 样式、使用 CSS 设计 XML 文档样式、设计响应式网站、设计酒店预订微信 wap 网站等内容。本书各章节注重实例间的联系和各功能间的难易层次，内容讲解以文字描述和图例并重，力求生动易懂，并对软件应用过程中的难点、重点和可能出现的问题给予详细讲解和提示。

除纸质内容外，本书还配备了多样化、全方位的学习资源，主要内容如下。

- ☑ **258 节同步教学微视频**
- ☑ **15000 项设计素材资源**
- ☑ **6 项拓展知识微阅读**
- ☑ **4800 个前端开发案例**
- ☑ **378 个实例案例分析**
- ☑ **48 本权威参考学习手册**
- ☑ **310 个在线微练习**
- ☑ **1036 道企业面试真题**

本书语言通俗易懂，突出实用性，案例丰富，结构清晰，图文并茂。本书可以作为网页设计与制作人员、商业网站建设与开发人员的自学用书，也可作为高等院校及相关培训机构的教学参考用书。

图书在版编目（CIP）数据

CSS3 网页设计从入门到精通：微课精编版/前端科技编著. —北京：清华大学出版社，2019
（清华社“视频大讲堂”大系 网络开发视频大讲堂）
ISBN 978-7-302-52249-2

Ⅰ. ①C…　Ⅱ. ①前…　Ⅲ. ①网页制作工具　Ⅳ. ①TP393.092.2

中国版本图书馆 CIP 数据核字（2019）第 018393 号

责任编辑： 贾小红
封面设计： 李志伟
版式设计： 魏　远
责任校对： 马子杰
责任印制： 沈　露

出版发行： 清华大学出版社
网　　址： http://www.tup.com.cn，http://www.wqbook.com
地　　址： 北京清华大学学研大厦 A 座　　**邮　　编：** 100084
社 总 机： 010-62770175　　**邮　　购：** 010-62786544
投稿与读者服务： 010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈： 010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者： 三河市铭诚印务有限公司
经　　销： 全国新华书店
开　　本： 203mm×260mm　　**印　　张：** 30.5　　**字　　数：** 922 千字
版　　次： 2019 年 4 月第 1 版　　**印　　次：** 2019 年 4 月第 1 次印刷
定　　价： 89.80 元

产品编号：079160-01

如何使用本书 

本书提供了多样化、全方位的学习资源，帮助读者轻松掌握 CSS3 网页设计技术，从小白快速成长为前端开发高手。



纸质书



视频讲解



拓展学习



在线练习



电子书

手机端+PC 端，线上线下同步学习

1．获取学习权限

学习本书前，请先刮开图书封底的二维码涂层，使用手机扫描，即可获取本书资源的学习权限。再扫描正文章节对应的 4 类二维码，可以观看视频讲解，阅读线上资源，查阅权威参考资料和在线练习提升，全程易懂、好学、速查、高效、实用。



2．观看视频讲解

对于初学者来说，精彩的知识讲解和透彻的实例解析能够引导其快速入门，轻松理解和掌握知识要点。本书中几乎所有案例都录制了视频，可以使用手机在线观看，也可以离线观看，还可以推送到计算机上大屏幕观看。







Note

3. 拓展线上阅读

一本书的厚度有限，但掌握一门技术却需要大量的知识积累。本书选择了那些与学习、就业关系紧密的核心知识点印在书中，而将大量的拓展性知识放在云盘上，读者扫描"线上阅读"二维码，即可免费阅读数百页的前端开发学习资料，获取大量的额外知识。



4. 进行线上练习

为方便读者巩固基础知识，提升实战能力，本书附赠了大量的前端练习题目。读者扫描最后一节的"在线练习"二维码，即可通过反复的实操训练加深对知识的领悟程度。

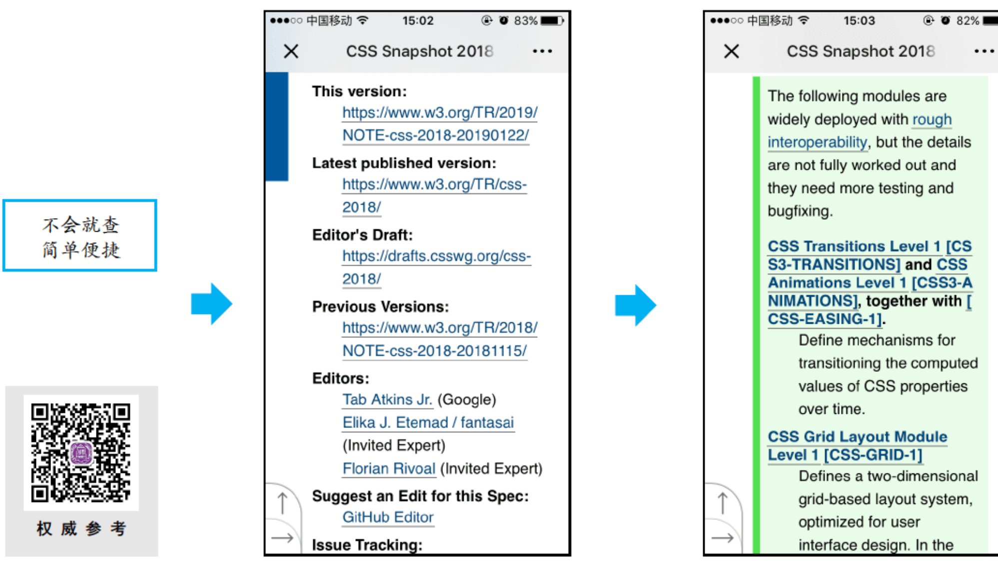


5. 查阅权威参考资料

扫描"权威参考"二维码，即可跳转到对应知识的官方文档上。通过大量查阅，真正领悟技术内涵。







6. 其他 PC 端资源下载方式

除了前面介绍过的可以直接将视频、拓展阅读等资源推送到邮箱之外，还提供了如下几种 PC 端资源获取方式。

☑ 登录清华大学出版社官方网站（www.tup.com.cn），在对应图书页面下查找资源的下载方式。

☑ 申请加入 QQ 群、微信群，获得资源的下载方式。

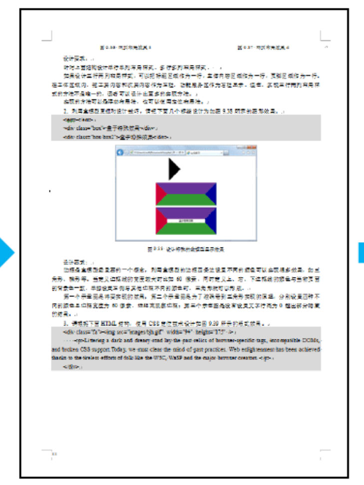
☑ 扫描图书封底“文泉云盘”二维码，获得资源的下载方式。

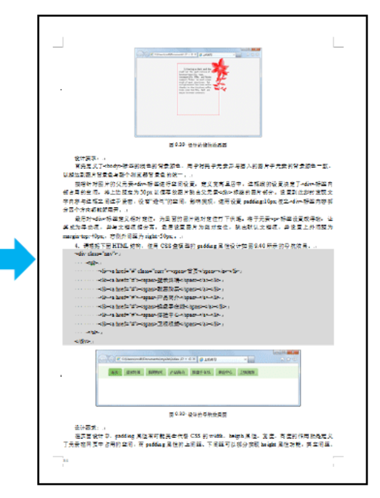
小白学习电子书

为方便读者全面提升，本书赠送了“前端开发百问百答”小白学习电子书。这些内容精挑细选，希望成为您学习路上的好帮手，关键时刻解您所需。

扫描小白手册封面的二维码，可在手机、平板上学习小白手册内容。











Note

从小白到高手的蜕变

谷歌的创始人拉里·佩奇说过，如果你刻意练习某件事超过10000个小时，那么你就可以达到世界级。

因此，不管您现在是怎样的前端开发小白，只要您按着下面的步骤来学习，假以时日，您会成为令自己惊讶的技术大咖。

（1）扎实的基础知识+大量的中小实例训练+有针对性地做一些综合案例。

（2）大量的项目案例观摩、学习、操练，塑造一定的项目思维。

（3）善于借用他山之石，对一些成熟的开源代码、设计素材拿来就用，学会站在巨人的肩膀上。

（4）有工夫多参阅一些官方权威指南，拓展自己对技术的理解和应用能力。

（5）最为重要的是，多与同行交流，在切磋中不断进步。

书本厚度有限，学习空间无限。纸张价格有限，知识价值无限。希望本书能帮您真正收获学习的乐趣和知识。最后，祝您阅读快乐！

前　言

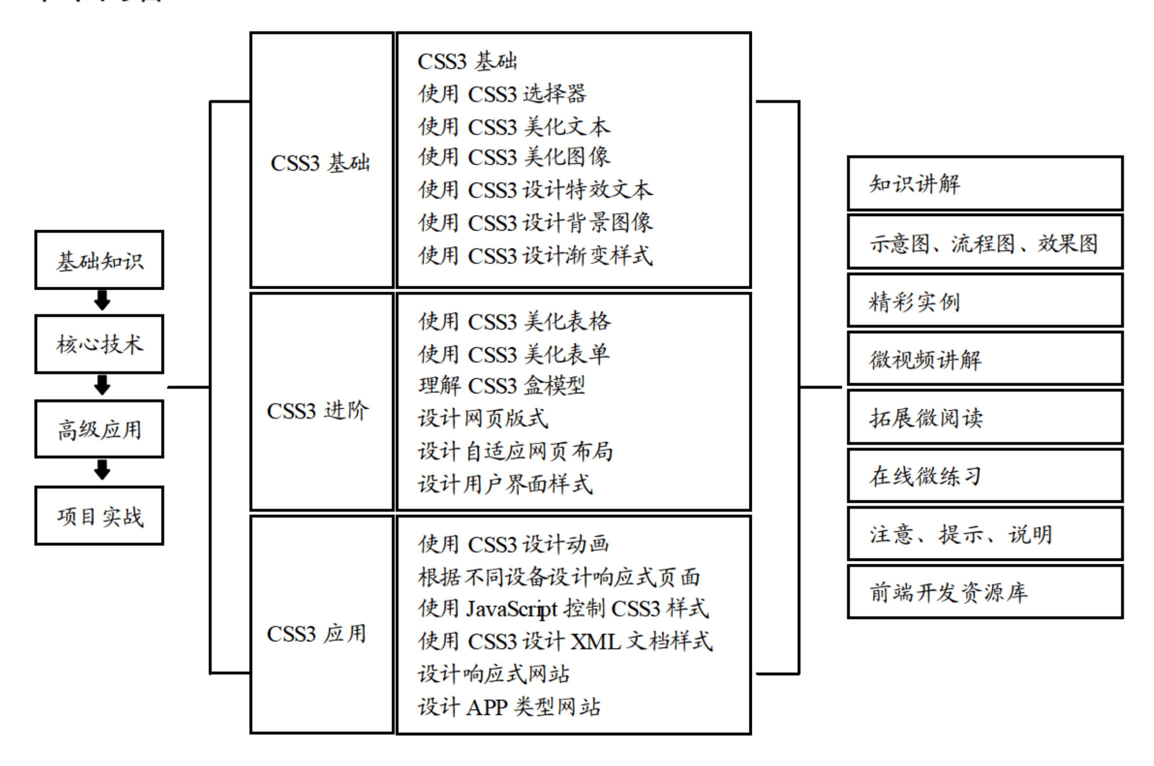
Preface

“网络开发视频大讲堂”系列丛书于 2013 年 5 月出版，因其编写细腻、讲解透彻、实用易学、配备全程视频等，备受读者欢迎。丛书累计销售近 20 万册，其中，《HTML5+CSS3 从入门到精通》累计销售 10 万册。同时，系列书被上百所高校选为教学参考用书。

本次改版，在继承前版优点的基础上，进一步对图书内容进行了优化，选择面试、就业最急需的内容，重新录制了视频，同时增加了许多当前流行的前端技术，提供了“入门学习→实例应用→项目开发→能力测试→面试”等各个阶段的海量开发资源库，实战容量更大，以帮助读者快速掌握前端开发所需要的核心精髓内容。

CSS3 是在 CSS2.1 基础上扩展而来的，本书将详细介绍各种有用的 CSS3 技术，总结 CSS3 设计中的最佳实践案例，讨论解决各种实际问题，以帮助开发者更好地掌握 CSS3 的特性，并且将新技术运用到实际开发中，提高自己开发 Web 程序的水平。

本书内容







本书特点

1. 由浅入深，编排合理，实用易学

本书系统地讲解了CSS3技术在网页设计中各个方面应用的知识，循序渐进，配合大量实例，帮助读者奠定坚实的理论基础。本书全面、细致地展示CSS3的基础知识，同时讲解了CSS3最实用、最流行的技术。

2. 跟着案例和视频学，入门更容易

跟着例子学习，通过训练提升，是初学者最好的学习方式。本书案例丰富详尽，且都附有详尽的代码注释及清晰的视频讲解。跟着这些案例边做边学，可以避免学到的知识流于表面、限于理论，尽情感受编程带来的快乐和成就感。

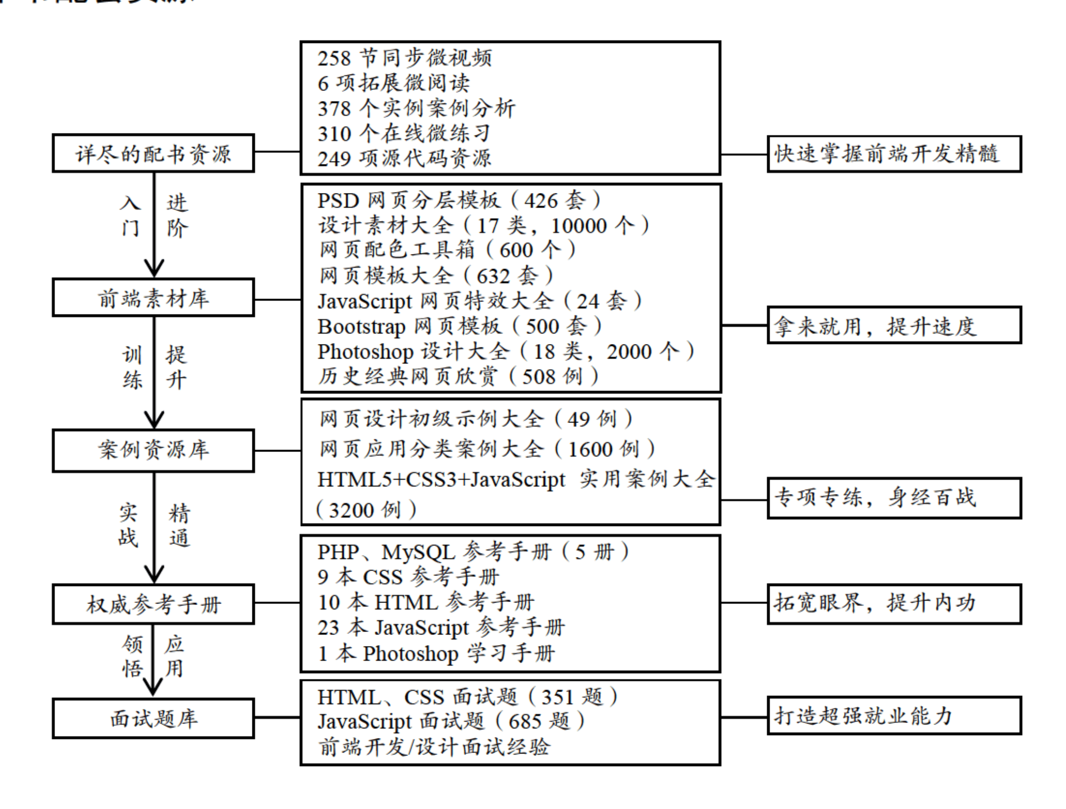
3. 4 大类线上资源，多元化学习体验

为了传递更多知识，本书力求突破传统纸质书的厚度限制。本书提供了4大类线上微资源，通过手机扫码，读者可随时观看讲解视频，拓展阅读相关知识，在线练习强化提升，还可以查阅官方权威资料，全程便捷、高效，感受不一样的学习体验。

4. 精彩栏目，易错点、重点、难点贴心提醒

本书根据初学者特点，在一些易错点、重点、难点位置精心设置了“注意”“提示”等小栏目。通过这些小栏目，读者会更留心相关的知识点和概念，绕过陷阱，掌握很多应用技巧。

本书配套资源







读者对象

☑ 具有一定网站开发经验的初、中级用户。
☑ 立志从事网站开发工作的从业人员。
☑ 自学网页设计或网站开发的大中专学生。
☑ 各类网站站长。
☑ 本书也可以作为各大中专院校相关专业的教学辅导和参考用书，或作为相关培训机构的培训教材。

读前须知

本书所有 HTML 示例都应该嵌套在一个有效文档的<body>标签中。同时，CSS 包含在内部或外部样式表中。对于包含重复内容的 HTML 示例，本书可能不会列出每一行代码，而是适时地使用省略号表示部分代码，详细代码需要参阅本书配套的示例源码。

本书所列出的插图可能会与读者实际环境中的操作界面有所差别，这可能是由于操作系统平台、浏览器版本等不同而引起的，在此特别说明，读者应该以实际情况为准。

本书所有案例代码都是在 HTML5 类型文档中编写的，所有示例也同样兼容 HTML 4.01 和 XHTML 1.0。

由于 CSS3 技术在不断地完善与更新中，建议根据本书提供的参考地址，获取有关 CSS3 最新的信息与服务。

读者服务

学习本书时，请先扫描封底的权限二维码（需要刮开涂层）获取学习权限，然后即可免费学习书中的所有线上线下资源。

本书所附赠的超值资源库内容，读者可登录清华大学出版社网站（www.tup.com.cn），在对应图书页面下获取其下载方式。也可扫描图书封底的“文泉云盘”二维码，获取其下载方式。

本书提供 QQ 群（668118468、697651657）、微信公众号（qianduankaifa_cn）、服务网站（www.qianduankaifa.cn）等互动渠道，提供在线技术交流、学习答疑、技术资讯、视频课堂、在线勘误等功能。在这里，您可以结识大量志同道合的朋友，在交流和切磋中不断成长。

读者对本书有什么好的意见和建议，也可以通过邮箱（qianduanjiaoshi@163.com）发邮件给我们。

关于作者

前端科技是由一群热爱 Web 开发的青年骨干教师和一线资深开发人员组成的一个团队，主要从事 Web 开发、教学和培训。参与本书编写的人员包括咸建勋、奚晶、文菁、李静、钟世礼、袁江、甘桂萍、刘燕、杨凡、朱砚、余乐、邹仲、余洪平、谭贞军、谢党华、何子夜、赵美青、牛金鑫、孙玉静、左超红、蒋学军、邓才兵、陈文广、李东博、林友赛、苏震巍、崔鹏飞、李斌、郑伟、邓艳超、胡晓霞、朱印宏、刘望、杨艳、顾克明、郭靖、朱育贵、刘金、吴云、赵德志、张卫其、李德光、刘坤、彭方强、雷海兰、王鑫铭、马林、班琦、蔡霞英、曾德剑等。

尽管已竭尽全力，但由于水平有限，书中疏漏和不足之处在所难免，欢迎各位读者朋友批评、指正。

编　者
2019 年 1 月

目　录



Contents




Note




Note



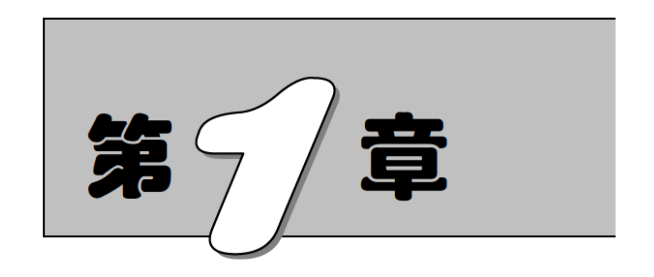
Note








Note



CSS3 基础

CSS3 是 CSS 规范的最新版本，在 CSS2 基础上增加了很多新功能，如圆角、多背景、透明度、阴影等，以帮助开发人员解决一些实际问题。本章将简单介绍 CSS 基础知识，读者可初步了解 CSS 的基本用法。

【学习重点】

⏭ 了解 CSS。

⏭ 正确使用 CSS。

⏭ 了解 CSS 基本特性。



1.1　CSS概述



本节简单介绍一下CSS的基本情况。

1.1.1　CSS历史

早期的HTML结构和样式是混在一起的，通过HTML标签组织内容，通过标签属性设置显示效果，这就造成了网页代码混乱不堪，代码维护非常困难。

1994年年初，哈坤•利提出了CSS的最初建议。伯特•波斯（Bert Bos）当时正在设计一款Argo浏览器，于是他们一拍即合，决定共同开发CSS。

1994年年底，哈坤在芝加哥的一次会议上第一次展示了CSS的建议，1995年他与波斯一起再次展示这个建议。当时W3C（World Wide Web Consortium，万维网联盟）组织刚刚成立，W3C对CSS的前途很感兴趣，为此组织了一次讨论会，哈坤、波斯是这个项目的主要技术负责人。

1996年年底，CSS语言正式设计完成，同年12月，CSS的第一个版本正式出版（http://www.w3.org/TR/CSS1/）。

权威参考 1

1997年年初，W3C组织专门负责CSS的工作组，负责人是克里斯•里雷。于是该工作组开始讨论第一个版本中没有涉及的问题。

1998年5月，CSS2正式出版（http://www.w3.org/TR/CSS2/）。

2002年，W3C的CSS工作组启动了CSS2.1开发。CSS2.1是CSS2的修订版，它纠正了CSS 2.0版本中的一些错误，并且更精确地描述CSS的浏览器实现。

权威参考 2

2004年，CSS2.1正式发布。

2006年年底，CSS2.1进一步完善，CSS2.1也成为目前最流行、获得浏览器支持最完整的CSS版本，它更准确地反映了CSS当前的状态。

CSS3的开发工作在2000年之前就开始了，但是距离最终的发布还有相当长的路要走，为了提高开发速度，也为了方便各主流浏览器根据需要渐进式支持，CSS3按模块化进行全新设计，这些模块可以独立发布和实现，这也为日后CSS的扩展奠定了基础。

到目前为止，CSS3还没有推出正式的完整版，但是已经陆续推出了不同的模块，这些模块已经被大部分浏览器支持或部分实现。

CSS3属性支持情况请访问https://bestvpn.org/whats-my-ip/详细了解。可以看出，完全支持CSS3属性的浏览器包括Chrome和Safari，其他主流浏览器除IE早期版本和Firefox 3外也基本支持。


权威参考 3

1.1.2　CSS3模块

CSS1和CSS2.1都是单一的规范，其中CSS1主要定义了网页对象的基本样式，如字体、颜色、背景、边框等。CSS2增加了高级概念：浮动、定位，以及高级选择器，如子选择器、相邻选择器和通用选择器等。


权威参考 1

CSS3被划分成多个模块组，每个模块组都有自己的规范。这样的好处是CSS3规范的发布不会因为存在争论的部分而影响其他模块的推进。对于浏览器来说，可以根据需要决定哪些CSS功能被支持。对于W3C制定者来说，可以根据需要进行针对性的更新，从而更加灵活和及时地修订一个整体的规范，这样更容易扩展新鲜的技术特性。


权威参考 2

2001年5月23日，W3C完成CSS3的工作草案，在该草案中制订了CSS3发展路



线图，路线图详细列出了所有模块，并计划在未来逐步进行规范（权威参考：http://www.w3.org/TR/css3-roadmap/）。

下面简单介绍 CSS3 各主要模块内容和参考地址，用户仅做了解或备查参考。



☑ 2002 年 5 月 15 日发布 CSS3 line 模块（http://www.w3.org/TR/CSS3-linebox/），该模块规范了文本行模型。

☑ 2002 年 11 月 7 日发布 CSS3 Lists 模块（http://www.w3.org/TR/CSS3-lists/），该模块规范了列表样式。

☑ 2002 年 11 月 7 日发布 CSS3 Border 模块（http://www.w3.org/TR/2002/WDcss3-border-20021107/），新添加了背景边框功能。该模块后来合并到背景模块（http://www.w3.org/TR/css3-background/）中。

☑ 2003 年 5 月 14 日发布 CSS3 Generated and Replaced Content 模块（http://www.w3.org/TR/css3-content/），该模块定义了 CSS3 的生成及更换内容功能。

☑ 2003 年 8 月 13 日发布 CSS3 Presentation Levels 模块（http://www.w3.org/TR/css3-preslev/），该模块定义了演示效果功能。

☑ 2003 年 8 月 13 日发布 CSS3 Syntax 模块（http://www.w3.org/TR/CSS3-syntax/），该模块重新定义了 CSS 语法规则。

☑ 2004 年 2 月 24 日发布 CSS3 Hyperlink Presentation 模块（http://www.w3.org/TR/css3-hyperlinks/），该模块重新定义了超链接表示规则。

☑ 2004 年 12 月 16 日发布 CSS3 S peech 模块（http://www.w3.org/TR/CSS3-speech/），该模块重新定义了语音“样式”规则。

☑ 2005 年 12 月 15 日发布 CSS3 Cascading and inheritance 模块（http://www.w3.org/TR/css3-cascade/），该模块重新定义了 CSS 层叠和继承规则。

☑ 2007 年 8 月 9 日发布 CSS3 Basic Box 模块（http://www.w3.org/TR/CSS3-box/），该模块重新定义了 CSS 基本盒模型规则。

☑ 2007 年 9 月 5 日发布 CSS3 Grid Positioning 模块（http://www.w3.org/TR/CSS3-grid/），该模块定义了 CSS 网格定位规则。

☑ 2009 年 3 月 20 日发布 CSS3 Animations 模块（http://www.w3.org/TR/CSS3-animations/），该模块定义了 CSS 动画模型。

☑ 2009 年 3 月 20 日发布 CSS3 3D Transforms 模块(http://www.w3.org/TR/CSS3-3d-transforms/)，该模块定义了 CSS 3D 转换模型。

☑ 2009 年 6 月 18 日发布 CSS3 Fonts 模块（http://www.w3.org/TR/CSS3-fonts/），该模块定义了 CSS 字体模型。

☑ 2009 年 7 月 23 日发布 CSS3 Image Values 模块（http://www.w3.org/TR/CSS3-images/），该模块定义了图像内容显示模型。

☑ 2009 年 7 月 23 日发布 CSS3 Flexible Box Layout 模块（http://www.w3.org/TR/CSS3-flexbox/），该模块定义了灵活的框布局模块。

☑ 2009 年 8 月 4 日发布 Lssom View uodule 模块（http://www.w3.org/TR/cssom-view/），该模块定义了 CSS 视图模块。

☑ 2009 年 12 月 1 日发布 CSS3 Transitions 模块（http://www.w3.org/TR/css-transitions-1/），该模块定义了动画过渡效果模型。

☑ 2009 年 12 月 1 日发布 CSS3 2D Transforms 模块（http://www.w3.org/TR/css-transforms-1/），



Note

该模块定义了 2D 转换模型。

☑ 2010 年 4 月 29 日发布 CSS3 Template Layout 模块（http://www.w3.org/TR/CSS3-layout/），该模块定义了模板布局模型。

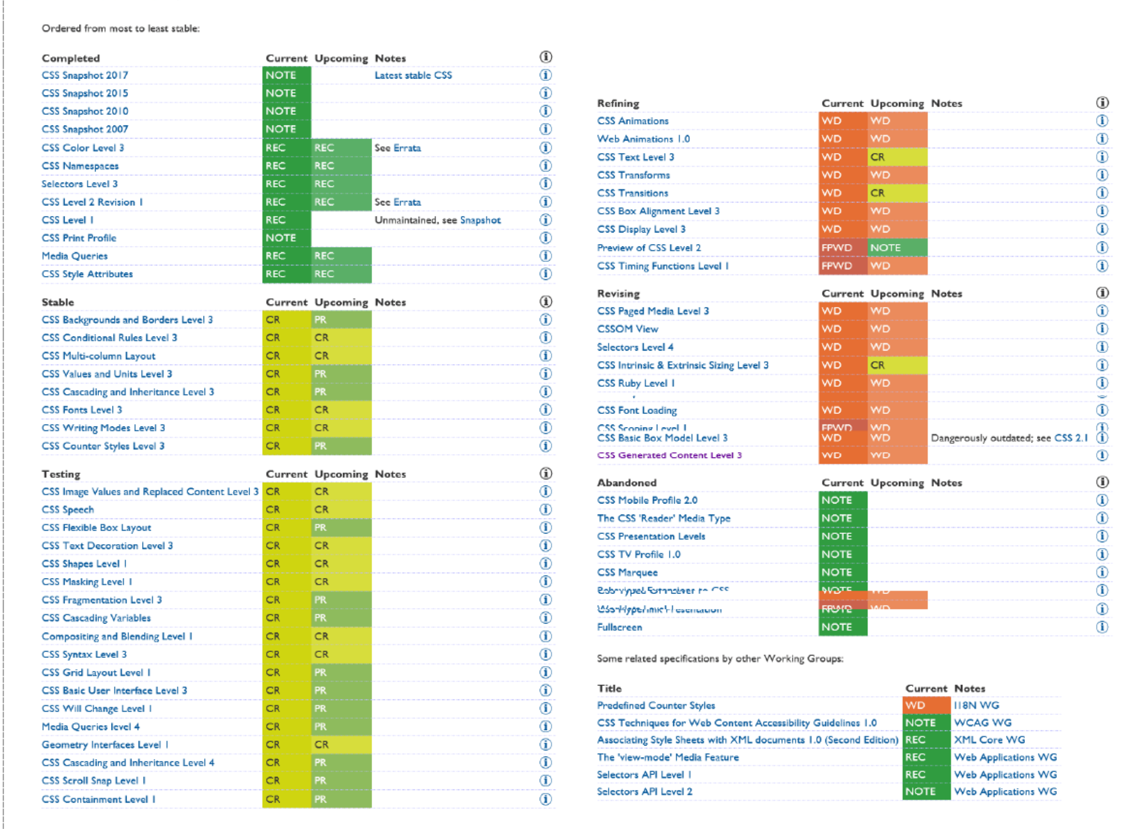
☑ 2010 年 4 月 29 日发布 CSS3 Generated Content for Paged Media 模块（http://www.w3.org/TR/css3-gcpm/），该模块定义了分页媒体内容模型。

☑ 2010 年 10 月 5 日发布 CSS3 Text 模块（http://www.w3.org/TR/CSS3-text/），该模块定义了文本模型。

☑ 2010 年 10 月 5 日发布 CSS3 Backgrounds and Borders 模块（http://www.w3.org/TR/CSS3-background/），该模块更新了边框和背景模型。

1.1.3 CSS3 开发状态

CSS3 每一个模块都有独立的开发和更新计划，如图 1.1 所示，从该图中可以看到 CSS3 当前发展的详细进度。更详细的信息可参见 http://www.w3.org/Style/CSS/current-work.html，其中介绍了 CSS3 具体划分为多少个模块组、CSS3 所有模块组目前所处的状态，以及将在什么时间发布。



Ordered from most to least stable:

Completed	Current	Upcoming	Notes
CSS Snapshot 2017	NOTE		Latest stable CSS
CSS Snapshot 2015	NOTE		
CSS Snapshot 2010	NOTE		
CSS Snapshot 2007	NOTE		
CSS Color Level 3	REC	REC	See Errata
CSS Namespaces	REC	REC	
Selectors Level 3	REC	REC	
CSS Level 2 Revision 1	REC	REC	See Errata
CSS Level 1	REC		Unmaintained, see Snapshot
CSS Print Profile	NOTE		
Media Queries	REC	REC	
CSS Style Attributes	REC	REC	

Stable	Current	Upcoming	Notes
CSS Backgrounds and Borders Level 3	CR	PR	
CSS Conditional Rules Level 3	CR	CR	
CSS Multi-column Layout	CR	CR	
CSS Values and Units Level 3	CR	PR	
CSS Cascading and Inheritance Level 3	CR	PR	
CSS Fonts Level 3	CR	CR	
CSS Writing Modes Level 3	CR	CR	
CSS Counter Styles Level 3	CR	PR	

Testing	Current	Upcoming	Notes
CSS Image Values and Replaced Content Level 3	CR	CR	
CSS Speech	CR	CR	
CSS Flexible Box Layout	CR	PR	
CSS Text Decoration Level 3	CR	CR	
CSS Shapes Level 1	CR	CR	
CSS Masking Level 1	CR	CR	
CSS Fragmentation Level 3	CR	PR	
CSS Cascading Variables	CR	PR	
Compositing and Blending Level 1	CR	CR	
CSS Syntax Level 3	CR	CR	
CSS Grid Layout Level 1	CR	PR	
CSS Basic User Interface Level 3	CR	PR	
CSS Will Change Level 1	CR	PR	
Media Queries level 4	CR	PR	
Geometry Interfaces Level 1	CR	CR	
CSS Cascading and Inheritance Level 4	CR	PR	
CSS Scroll Snap Level 1	CR	PR	
CSS Containment Level 1	CR	PR	

Refining	Current	Upcoming	Notes
CSS Animations	WD	WD	
Web Animations 1.0	WD	WD	
CSS Text Level 3	WD	CR	
CSS Transforms	WD	WD	
CSS Transitions	WD	CR	
CSS Box Alignment Level 3	WD	WD	
CSS Display Level 3	WD	WD	
Preview of CSS Level 2	FPWD	NOTE	
CSS Timing Functions Level 1	FPWD	WD	

Revising	Current	Upcoming	Notes
CSS Paged Media Level 3	WD	WD	
CSSOM View	WD	WD	
Selectors Level 4	WD	WD	
CSS Intrinsic & Extrinsic Sizing Level 3	WD	CR	
CSS Ruby Level 1	WD	WD	
CSS Font Loading	WD	WD	
CSS Scoping Level 1	FPWD	WD	
CSS Basic Box Model Level 3	WD	WD	Dangerously outdated; see CSS 2.1
CSS Generated Content Level 3	WD	WD	

Abandoned	Current	Upcoming	Notes
CSS Mobile Profile 2.0	NOTE		
The CSS 'Reader' Media Type	NOTE		
CSS Presentation Levels	NOTE		
CSS TV Profile 1.0	NOTE		
CSS Marquee	NOTE		
[illegible]	[illegible]	[illegible]	
[illegible]	[illegible]	[illegible]	
Fullscreen	NOTE		

Some related specifications by other Working Groups:

Title	Current	Notes
Predefined Counter Styles	WD	I18N WG
CSS Techniques for Web Content Accessibility Guidelines 1.0	NOTE	WCAG WG
Associating Style Sheets with XML documents 1.0 (Second Edition)	REC	XML Core WG
The 'view-mode' Media Feature	REC	Web Applications WG
Selectors API Level 1	REC	Web Applications WG
Selectors API Level 2	NOTE	Web Applications WG

图 1.1　CSS3 所有模块进度表

其中，Current 列表示模块当前的状态，Upcoming 列表示即将进行的状态。各种状态缩写词说明如下。

☑ WD：Working Draft，表示工作草案。



- ☑ LC：Last Call，表示最终工作草案。
- ☑ CR：Candidate Recommendation，表示候选推荐标准。
- ☑ PR：Proposed Recommendation，表示建议推荐标准。
- ☑ REC：Recommendation，表示推荐标准。



Note

【扩展】

W3C 标准只是推荐标准，并没有强制执行的效力。不过，鉴于 W3C 在 Web 标准领域的影响力和强大号召力，W3C 发布的推荐标准，通常浏览器厂商都很重视并积极支持。

一般情况下，W3C 标准制订经历下面几个阶段，这些阶段都有专有术语，拥有定义好的含义，虽然也有变化，但修订频率不高，最新版是 2005 年制订的，具体说明如下。

第 1 阶段：工作草案（Working Draft）。

工作组依据工作组章程（Charter）提出一系列工作草案。公众和 W3C 会员可以提出评论和问题，工作组必须处理这些反馈。本阶段时长依多种因素而变。

第 2 阶段：最终工作草案（Last Call Working Draft）。

工作组已完成工作，并要求公众和 W3C 会员提交最后的评论与问题。同样，工作组必须处理这些反馈。如果出现问题，可能要回到工作草案阶段。本阶段时长通常为 3 周，但也可以更长。

第 3 阶段：候选推荐标准（Candidate Recommendation）。

当最终工作草案阶段结束且问题都得到解决后，将进入候选推荐标准阶段。此时认为该规范已经稳定，可以展开试验性实施了。工作组必须将实施中得到的反馈整合到规范中。同样，如果出现问题，须返回工作草案阶段。根据实施进展，本阶段通常持续 0～1 年。

第 4 阶段：建议推荐标准（Proposed Recommendation）。

如无意外，规范将进入建议推荐标准阶段。在此阶段，W3C 总监（Tim Berners-Lee）将正式请求 W3C 会员审阅建议推荐标准。本阶段时长不少于 4 周。

第 5 阶段：推荐标准（Recommendation）。

根据审阅结果，W3C 总监宣布该规范成为 W3C 推荐标准，中间可能发生微小改动，或者返回工作草案阶段，或者彻底从 W3C 工作日程上移去。技术规范一旦成为推荐标准，它就是官方的 W3C 标准了。

当然由于种种因素，有些 W3C 草案未能在 W3C 得到青睐，最终只能成为 Note，这意味着没有厂商会去实现它。

最后，在实际操作中，很多浏览器厂商出于利益或技术上的考虑，可能会不完全遵照 W3C 推荐标准来实现其产品，因此用户会发现各个厂商的浏览器对 CSS3 技术支持程度各不相同。

1.1.4　浏览器支持状态

CSS3 特性大部分都已经有了很好的浏览器支持度。各主流浏览器对 CSS3 的支持越来越完善，下面来了解一下两大平台（Mac 和 Windows）、五大浏览器（Chrome、Firefox、Safari、Opera 和 IE）对 CSS3 新特性和 CSS3 选择器的支持情况。

CSS3 属性支持情况如图 1.2 所示（http://fmbip.com/litmus/）。可以看出，完全支持 CSS3 属性的浏览器有 Chrome 和 Safari，而且不管是 Mac 平台还是 Windows 平台全支持。

CSS3 选择器支持情况如图 1.3 所示（http://fmbip.com/litmus/）。除了 IE 家族和 Firefox 3，其他几乎全部支持。其中，Chrome、Safari、Firefox 3.6、Opera 10.5 最好。







平台	MAC				WIN								
浏览器	CHROME	FIREFOX	OPERA	SAFARI	CHROME	FIREFOX		OPERA		SAFARI	IE		
版本	5	3.6	10.1	4	4	3.6	3	10	10.5	4	6	7	8
RGBA	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗
HSLA	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗
Multiple Backgrounds	✓	✓	✗	✓	✓	✓	✗	✗	✓	✓	✗	✗	✗
Border Image	✓	✓	✗	✓	✓	✓	✗	✗	✓	✓	✗	✗	✗
Border Radius	✓	✓	✗	✓	✓	✓	✓	✗	✓	✓	✗	✗	✗
Box Shadow	✓	✓	✗	✓	✓	✓	✗	✗	✓	✓	✗	✗	✗
Opacity	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗
CSS Animations	✓	✗	✗	✓	✓	✗	✗	✗	✗	✓	✗	✗	✗
CSS Columns	✓	✓	✗	✓	✓	✓	✓	✗	✗	✓	✗	✗	✗
CSS Gradients	✓	✓	✗	✓	✓	✓	✗	✗	✗	✓	✗	✗	✗
CSS Reflections	✓	✗	✗	✓	✓	✗	✗	✗	✗	✓	✗	✗	✗
CSS Transforms	✓	✓	✗	✓	✓	✓	✗	✗	✓	✓	✗	✗	✗
CSS Transforms 3D	✓	✗	✗	✓	✓	✗	✗	✗	✗	✓	✗	✗	✗
CSS Transitions	✓	✗	✗	✓	✓	✗	✗	✗	✓	✓	✗	✗	✗
CSS FontFace	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓

图 1.2　CSS3 属性支持列表

平台	MAC				WIN								
浏览器	CHROME	FIREFOX	OPERA	SAFARI	CHROME	FIREFOX		OPERA		SAFARI	IE		
版本	5	3.6	10.1	4	4	3.6	3	10	10.5	4	6	7	8
CSS3: Begins with	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓
CSS3: Ends with	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓
CSS3: Matches	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓
CSS3: Root	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗
CSS3: nth-child	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✗	✗	✗
CSS3: nth-last-child	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✗	✗	✗
CSS3: nth-of-type	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✗	✗	✗
CSS3: nth-last-of-type	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✗	✗	✗
CSS3: last-child	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗
CSS3: first-of-type	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✗	✗	✗
CSS3: last-of-type	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✗	✗	✗
CSS3: only-child	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗
CSS3: only-of-type	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✗	✗	✗
CSS3: empty	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗
CSS3: target	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗
CSS3: enabled	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗
CSS3: disabled	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗
CSS3: checked	✓	✓	✗	✓	✓	✓	✓	✗	✓	✓	✗	✗	✗
CSS3: not	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗
CSS3: General Sibling	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓

图 1.3　CSS3 选择器支持列表





注意：各主流浏览器也定义大量私有属性，方便用户体验 CSS3 的新特性。例如：

☑ 以 Webkit 引擎为核心的浏览器（如 Safari、Chrome）的私有属性都以-webkit-前缀定义。
☑ 以 Gecko 引擎为核心的浏览器（如 Firefox）的私有属性都以-moz-前缀定义。
☑ 以 Konqueror 引擎为核心的浏览器的私有属性都以-khtml-前缀定义。
☑ Opera 浏览器的私有属性以-o-前缀定义。
☑ Internet Explorer 浏览器的私有属性以-ms-前缀定义，IE 8+支持-ms-前缀。

1.2　CSS 基本用法

CSS 是一种标识语言，可以在任何文本编辑器中编辑。下面简单介绍 CSS 的基本用法。



视频讲解

1.2.1　CSS 样式

CSS 的语法单元是样式，每个样式包含两部分内容：选择器和声明（或称为规则），如图 1.4 所示。

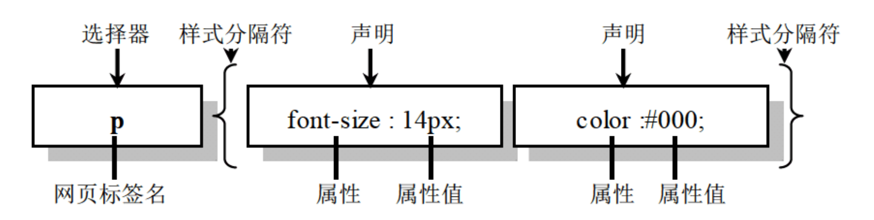


图 1.4　CSS 样式基本格式

☑ 选择器（Selector）：指定样式作用于哪些对象，这些对象可以是某个标签、指定 Class 或 ID 值的元素等。浏览器在解析这个样式时，根据选择器来渲染对象的显示效果。
☑ 声明（Declaration）：指定浏览器如何渲染选择器匹配的对象。声明包括属性和属性值两部分，并用分号来标识一个声明的结束，在一个样式中最后一个声明可以省略分号。所有声明被放置在一对大括号内，位于选择器的后面。
☑ 属性（Property）：CSS 预设的样式选项。属性名由一个单词或多个单词组成，多个单词之间通过连字符相连，这样能够很直观地了解属性所要设置样式的类型。
☑ 属性值（Value）：定义显示效果的值，包括值和单位；或者仅定义一个关键字。

【示例】下面示例简单演示了如何在网页中设计 CSS 样式。

【操作步骤】

第 1 步，启动 Dreamweaver，新建一个网页，保存为 test.html。

第 2 步，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表。

第 3 步，在<style>标签内输入下面样式代码，定义网页字体大小为 24px，字体颜色为白色。

```
body{font-size: 24px; color: #fff;}
```

第 4 步，输入下面样式代码，定义段落文本的背景色为蓝色。

```
p {background-color: #00F;}
```

第 5 步，在<body>标签内输入下面一段话，然后在浏览器中预览，效果如图 1.5 所示。



```
<body>
<p>莫等闲、白了少年头，空悲切。 </p>
</body>
```



Note

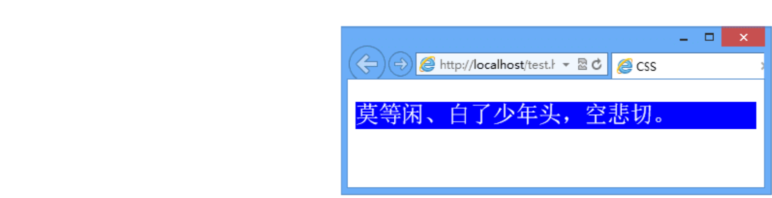


图 1.5　使用 CSS 定义段落文本样式



视频讲解

1.2.2　引入 CSS 样式

在网页中，有 3 种方法可以正确引入 CSS 样式，让浏览器能够识别和解析。

1．行内样式

把 CSS 样式代码置于标签的 style 属性中，例如：

```
<span style="color:red;">红色字体</span>
<div style="border:solid 1px blue; width:200px; height:200px;"></div>
```

这种用法没有真正把 HTML 结构与 CSS 样式分离，一般不建议大规模使用。除非为页面中某个元素临时设置特定样式。

2．内部样式

把 CSS 样式代码放在<style>标签内，例如：

```
<style type="text/css">
body {/*页面基本属性*/
    font-size: 12px;
    color: #CCCCCC;
}
/*段落文本基础属性*/
p {background-color: #FF00FF;}
</style>
```

这种用法也称为网页内部样式，适合为单页面定义 CSS 样式，不适合为一个网站或多个页面定义样式。

内部样式一般位于网页的头部区域，目的是让 CSS 源代码早于页面源代码下载并被解析，避免当网页下载之后还无法正常显示。

3．外部样式

把样式放在独立的文件中，然后使用<link>标签或者@import 关键字导入。一般网站都采用这种方法来设计样式，真正实现 HTML 结构和 CSS 样式的分离，以便统筹规划、设计、编辑和管理 CSS 样式。



视频讲解

1.2.3　CSS 样式表

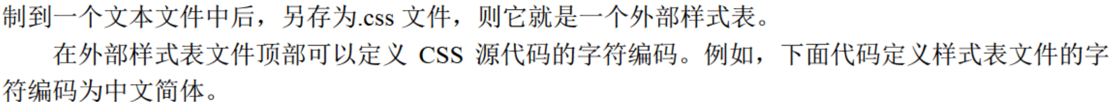
样式表是一个或多个 CSS 样式组成的样式代码段，分为内部样式表和外部样式表，它们没有本



质区别，只是存放位置不同。

内部样式表包含在<style>标签内，一个<style>标签就表示一个内部样式表。而通过标签的 style 属性定义的样式属性就不是样式表。如果一个网页文档中包含多个<style>标签，就表示该文档包含多个内部样式表。





如果 CSS 样式被放置在网页文档外部的文件中，则称为外部样式表，一个 CSS 样式表文档就表示一个外部样式表。实际上，外部样式表也就是一个文本文件，其扩展名为.css。当把不同的样式复制到一个文本文件中后，另存为.css 文件，则它就是一个外部样式表。

在外部样式表文件顶部可以定义 CSS 源代码的字符编码。例如，下面代码定义样式表文件的字符编码为中文简体。

```
@charset "gb2312";
```

如果不设置 CSS 文件的字符编码，可以保留默认设置，则浏览器会根据 HTML 文件的字符编码来解析 CSS 代码。

1.2.4　导入外部样式表



外部样式表文件可以通过两种方法导入 HTML 文档中。

1．使用<link>标签

使用<link>标签导入外部样式表文件的代码如下：

```
<link href="001.css" rel="stylesheet" type="text/css" />
```

该标签必须设置的属性说明如下。

☑　href：定义样式表文件 URL。

☑　type：定义导入文件类型，同 style 元素一样。

☑　rel：用于定义文档关联，这里表示关联样式表。

2．使用@import 命令

在<style>标签内使用@import 关键字导入外部样式表文件的方法如下：

```
<style type="text/css">
@import url("001.css");
</style>
```

在@import 关键字后面，利用 url()函数包含具体的外部样式表文件的地址。

1.2.5　CSS 格式化



在 CSS 中增加注释很简单，所有被放在“/*”和“*/”分隔符之间的文本信息都被称为注释。例如：

```
/*注释*/
```

或

```
/*
注释
*/
```



在 CSS 中，各种空格是不被解析的，因此用户可以利用 Tab 键、空格键对样式表和样式代码进行格式化排版，以方便阅读和管理。

1.2.6 CSS 属性





视频讲解



视频讲解

CSS 属性众多，在 W3C CSS 2.0 版本中共有 122 个标准属性（http://www.w3.org/TR/CSS2/propidx.html），在 W3C CSS 2.1 版本中共有 115 个标准属性（http://www.w3.org/TR/CSS21/propidx.html），其中删除了 CSS 2.0 版本中 7 个属性：font-size-adjust、font-stretch、marker-offset、marks、page、size 和 text-shadow。在 W3C CSS 3.0 版本中又新增加了 20 多个属性（http://www.w3.org/Style/CSS/current-work#CSS3）。

属性的用法将在后面各章节中详细说明，用户也可以通过 CSS3 参考手册具体了解。

1.2.7 CSS 属性值

CSS 属性取值比较多，具体类型包括长度、角度、时间、频率、布局、分辨率、颜色、文本、函数、生成内容、图像和数字。下面重点介绍一下常用的长度值，其他类型值将在相应属性中具体说明。

长度值包括绝对值和相对值两类。

1．绝对值

绝对值在网页中很少使用，一般用在特殊的场合。常见绝对单位如下。

- ☑ 英寸（in）：使用最广泛的长度单位。
- ☑ 厘米（cm）：最常用的长度单位。
- ☑ 毫米（mm）：在研究领域使用广泛。
- ☑ 磅（pt）：也称点，在印刷领域使用广泛。
- ☑ pica（pc）：在印刷领域使用。

2．相对值

相对值是根据屏幕分辨率、可视区域、浏览器设置，以及相关元素的大小等因素确定值的大小。常见相对单位如下。

- ☑ em：表示字体高度，它能够根据字体的 font-size 值来确定大小，例如：

```
p{/*设置段落文本属性*/
    font-size:12px;
    line-height:2em;/*行高为 24px*/
}
```

从上面样式代码中可以看出：一个 em 等于 font-size 的属性值，如果设置 font-size:12pt，则 line-height:2em 就会等于 24pt。如果设置 font-size 属性的单位为 em，则 em 的值将根据父元素的 font-size 属性值来确定。例如，定义如下 HTML 局部结构：

```
<div id="main">
    <p>em 相对长度单位使用</p>
</div>
```

再定义如下样式：

```
#main {     font-size:12px;}
p {font-size:2em;} /*字体大小将显示为 24px*/
```



同理，如果父对象的 font-size 属性的单位也为 em，则将依次向上级元素寻找参考的 font-size 属性值，如果都没有定义，则会根据浏览器默认字体进行换算，默认字体一般为 16px。

☑ ex：表示字母 x 的高度。

☑ px：根据屏幕像素点来确定大小。这样不同的显示分辨率就会使相同取值的 px 单位所显示出来的效果截然不同。

☑ %：百分比也是一个相对单位值。百分比值总是通过另一个值来确定当前值，一般参考父对象中相同属性的值。例如，如果父元素宽度为 500px，子元素的宽度为 50%，则子元素的实际宽度为 250px。



1.3　CSS 特性

CSS 样式具有两个特性：继承性和层叠性，下面分别进行说明。

1.3.1　CSS 继承性



视 频 讲 解

CSS 继承性是指后代元素可以继承祖先元素的样式。继承样式主要包括字体、文本等基本属性，如字体、字号、颜色、行距等，对于下面类型属性是不允许继承的：边框、边界、补白、背景、定位、布局、尺寸等。

提示：灵活应用 CSS 继承性，可以优化 CSS 代码，但是继承的样式的优先级是最低的。

【示例】下面示例在 body 元素中定义整个页面的字体大小、字体颜色等基本页面属性，这样包含在 body 元素内的其他元素都将继承该基本属性，以实现页面显示效果的统一。

新建网页，保存为 test.html，在<body>标签内输入如下代码，设计一个多级嵌套结构。

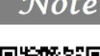
```
<div id="wrap">
    <div id="header">
        <div id="menu">
            <ul>
                <li><span>首页</span></li>
                <li>菜单项</li>
            </ul>
        </div>
    </div>
    <div id="main">
        <p>主体内容</p>
    </div>
</div>
```

在<head>标签内添加<style type="text/css">标签，定义内部样式表，然后为 body 定义字体大小为 12px，通过继承性，则包含在 body 元素中的所有其他元素都将继承该属性，并显示包含的字体大小为 12px。在浏览器中预览，显示效果如图 1.6 所示。

```
body {font-size:12px;}
```







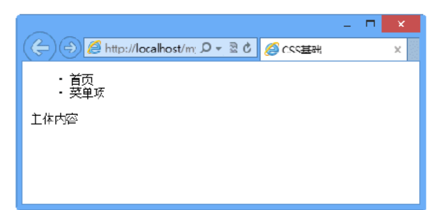


图 1.6　CSS 继承性演示效果

1.3.2　CSS 层叠性

CSS 层叠性是指 CSS 能够对同一个对象应用多个样式的能力。

【示例 1】新建一个网页，保存为 test.html，在<body>标签内输入如下代码：

```
<div id="wrap">看看我的样式效果</div>
```

在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，分别添加两个样式：

```
div {font-size:12px;}
div {font-size:14px;}
```

两个样式中都声明相同的属性，并应用于同一个元素上。在浏览器中测试，则会发现最后字体显示为 14px，也就是说 14px 字体大小覆盖了 12px 字体大小，这就是样式层叠。

当多个样式作用于同一个对象时，则根据选择器的优先级，确定对象最终应用的样式。

☑　标签选择器：权重值为 1。
☑　伪元素或伪对象选择器：权重值为 1。
☑　类选择器：权重值为 10。
☑　属性选择器：权重值为 10。
☑　ID 选择器：权重值为 100。
☑　其他选择器：权重值为 0，如通配选择器等。

然后，以上面权值数为起点来计算每个样式中选择器的总权值数。计算规则如下。

☑　统计选择器中 ID 选择器的个数，然后乘以 100。
☑　统计选择器中类选择器的个数，然后乘以 10。
☑　统计选择器中标签选择器的个数，然后乘以 1。

以此方法类推，把所有权重值数相加，即可得到当前选择器的总权重值，最后根据权重值来决定哪个样式的优先级大。

【示例 2】新建一个网页，保存为 test.html，在<body>标签内输入如下代码：

```
<div id="box" class="red">CSS 选择器的优先级</div>
```

在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，添加如下样式：

```
body div#box {border:solid 2px red;}
#box {border:dashed 2px blue;}
div.red {border:double 3px red;}
```

对于上面的样式表，可以这样计算它们的权重值：

body div#box = 1 + 1 + 100 = 102

#box = 100

di.red = 1 + 10 = 11



因此，最后的优先级为 body div#box 大于#box，#box 大于 di.red。所以可以看到显示效果为 2px 的红色实线。在浏览器中预览，显示效果如图 1.7 所示。

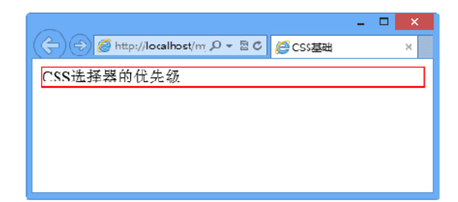


图 1.7　CSS 优先级的样式演示效果



提示： 与样式表中样式相比，行内样式优先级最高；相同权重值时，样式最近的优先级最高；使用!important 命令定义的样式优先级绝对高；!important 命令必须位于属性值和分号之间，如#header{color:Red!important;}，否则无效。

1.4　案 例 实 战

视 频 讲 解

本节设计一个完整页面，体验标准网页的制作过程。案例页面设计效果如图 1.8 所示。



图 1.8　使用 CSS 设计的第一个页面

【操作步骤】

第 1 步，启动 Dreamweaver，新建 HTML 文档，保存为 index.html。

提示： 本页面所需要的图片等素材可以参考附赠的源代码。考虑到很多初学者是第一次接触 CSS，本案例稍显复杂，因此建议读者可根据实际情况选择性地学习，或直接跳过本节操作练习。



第 2 步，切换到代码视图，在<body>标签内输入下面代码，构建本页面主体结构，设计本例页面一级框架。



```
<!--[一级框架]-->
<!--顶部-->
<div id="top"></div>
<div id="top1"></div>
<!--主体-->
<div id="main"></div>
<!--底部-->
<div id="footer"></div>
<div id="copyright"></div>
```

在标准布局中，读者应该为每个 div 框架元素定义 id 属性，这些 id 属性如同人的身份证一样，方便 CSS 准确地控制每个 div 布局块。所以，为了阅读和维护的需要，应该为 id 属性起一个有意义的名字。

第 3 步，进一步细化页面结构，设计页面内部层次框架。由于本例页面比较简单，嵌套框架不会很深，顶部和底部布局块可能就不要嵌套框架。输入完整的 HTML 结构代码：

```
<!--[完整 HTML 框架]-->
<!--顶部-->
<div id="top"></div>
<div id="top1"><img src="images/bg_top.jpg" width="776" height="121" /></div>
<!--主体-->
<div id="main">
    <div id="content">
        <div id="title">Hello World -- 第一个 CSS3+DIV 页面</div>
        <div class="sub">实例</div>
        <div class="box"><div class="tl"><div class="tr"><div class="bl"><div  class="content br">
```

第 4 步，丰富结构内容，使用<pre>标签显示代码内容，使用<a>设计超链接文本，整个页面内容显示如下，代码内容是在网页中居中显示红色字符 Hello World!。

```
<pre>
&lt;!doctype html&gt;
&lt;html&gt;
    &lt;head&gt;
        &lt;meta charset=&quot;utf-8&quot;&gt;
        &lt;title&gt;Hello World&lt;/title&gt;
        &lt;style type=&quot;text/css&quot;&gt;
        h1 {
            color: #FF0000;
            text-align: center;
        }
        &lt;/style&gt;
    &lt;/head&gt;
    &lt;body&gt;
        &lt;h1&gt;Hello World! &lt;/h1&gt;
    &lt;/body&gt;
&lt;/html&gt;
</pre>
```





```
                </div></div></div></div></div>
            <div id="gotop"><a title="跳到页首" href="#top">返回顶部</a></div>
        </div>
    </div>
    <!--底部-->
    <div id="footer"></div>
    <div id="copyright">
        &copy;2017 <a href="#" target="_black" >mysite.cn</a> all rights reserved
    </div>
```

上面所用的 HTML 框架代码嵌套层次只有 3 层，其中为了实现圆角区域的显示效果而单独嵌套的多层 div 元素除外。

第 5 步，按 Ctrl+S 快捷键保存文档，按 F12 键在浏览器中预览，则显示效果如图 1.9 所示。现在还没有定义 CSS 代码，所以这不是最终效果。



图 1.9　页面的 HTML 结构预览效果

第 6 步，编写 CSS 代码放在一个单独的文件中。新建 CSS 文档，保存为 style.css，文件扩展名为.css。

第 7 步，打开 index.html 文档，然后在<head>标签内部插入一个<link>标签，输入下面代码，导入第 6 步新建的外部样式表文件。

```
<!--[在网页中链接外部样式表文件]-->
<LINK href="images/style.css" type=text/css rel=stylesheet>
```

第 8 步，打开 style.css 文档，在其中输入下面 CSS 代码：

```
/*公共属性
-----------------------------------*/
html {min-width: 776px;}
/*页面属性：边距为 0，字体颜色为黑色，字体大小为 14px，行高为字体大小的 1.6 倍，居中对齐，背景色为天蓝色，字体为宋体等*/
body {margin: 0px; padding: 0px; border: 0px; color: #000; font-size: 14px; line-height: 160%; text-align: center; background: #6D89DD; font-family: '宋体','新宋体',arial,verdana,sans-serif;}
/*超链接属性：无边距、无边框、无下画线，然后定义正常状态下的颜色、访问过的颜色和鼠标经过时的颜
```



Note

```
色并显示下画线*/
    a {margin: 0px; padding: 0px; border: 0px; text-decoration: none;}
    a:link {color: #E66133;}
    a:visited{color: #E66133;}
    a:hover{color: #637DBC; text-decoration: underline;}
    /*预定义格式属性：无首行缩进，浅灰色背景，内边距和外边距为 0，字体颜色为蓝色*/
    pre {text-indent: 0; background: #DDDDDD; padding: 0; margin: 0; color: blue;}
    /*顶部布局
    -----------------------------------*/
    #top{width: 776px; margin-right: auto; margin-left: auto; padding: 0px; height: 12px; background: url(images/
bg_top1.gif) #fff repeat-x left top; overflow: hidden;}

    #top1{width: 776px; margin-right: auto; margin-left: auto; padding: 0px; height: 121px;}
    /* 主体布局
    -----------------------------------*/
    /*外层定义背景图像，实现麻点显示效果*/
    #main{width: 776px; margin-right: auto; margin-left: auto; padding: 1.2em 0px; background: url(images/
bg_dot1.gif) #fff repeat left top; text-align: left;}
    /*内层定义背景颜色为白色，实现中间内容区域遮盖麻点显示*/
    #content{width: 710px; margin-right: auto; margin-left: auto; padding: 1em; background: #fff;}
    /*大标题区域属性*/
    #title {font-weight: bold; margin: 0px 0px 0.5em 0px; padding: 0.5em 0px 0.5em 1em; font-size: 24px; color:
#00A06B; text-align: left; border-bottom: solid   #9EA3C1   2px;}
    /*小标题区域属性*/
    .sub {color: #00A06B; font-weight: bold; font-size: 13px; text-align: left; padding: 1em 2em 0; background:
url(images/0.gif) #fff no-repeat 1em 74%;}
    /*内容区域显示属性*/
    .content {text-indent: 2em; font-size: 13px; margin-left: 2em; padding: 1em 6px;}
    /*页内链接区域属性*/
    #gotop{width: 100%; margin: 0px; padding: 0px; background: #fff; height: 2em; font-size: 12px; text-align: right;}
    /*底部布局
    -----------------------------------*/
    /*页脚装修图*/
    #footer{clear: both; width: 776px; margin-right: auto; margin-left: auto; padding: 0px; background: url(images/
bg_bottom.gif) #fff repeat left top; text-align: center; height: 39px; color: #ddd;}
    /*版权信息*/
    #copyright{width: 776px; margin-right: auto; margin-left: auto; padding: 5px 0px 0px 0px; background: #fff;
text-align: center; height: 60px; line-height: 13px; font-size: 12px; color: #9EA0BB;}
    #copyright a {color: #667EBE;}
    /*圆角特效
    -----------------------------------*/
    .box {background: url(images/nt.gif) repeat;}
    .tl {background: url(images/tl.gif) no-repeat top left;}
    .tr {background: url(images/tr.gif) no-repeat top right;}
    .bl {background: url(images/bl.gif) no-repeat bottom left;}
    .br {background: url(images/br.gif) no-repeat bottom right;}
```

读者可能看不懂上面的 CSS 代码，只需根据上面的提示简单了解即可。其中，width 属性用来定义宽度；“background: url(images/bg_bottom.gif) #fff repeat left top;”规则用来定义背景图像重复铺展显示，其中 url 指定背景图像的地址，repeat 属性定义铺展显示，left top 表示背景图像的起始位置为左



上角。其他属性上面代码中已有解释，读者可以尝试阅读一下，如果读不懂也没有关系，毕竟现在仅是开始。相信随着学习的深入，一定会明白上面代码的意思。

另外，本节案例没有使用 CSS3 圆角属性定义区块圆角，而是使用传统方法进行设计，主要考虑初学者的学习水平，后面章节我们会详细介绍。

第 9 步，按 Ctrl+S 快捷键保存文档，然后在浏览器中再次预览页面，则可以看到最终效果。



Note

1.5　在 线 练 习

为了帮助读者打下扎实的 CSS 基础，本节提供一些课外练习实例，感兴趣的读者可以扫码练习。



在 线 练 习



第2章 CSS3 选择器

CSS3 选择器在 CSS2.1 选择器的基础上新增了部分属性选择器和伪类选择器，减少对 HTML 类和 ID 的依赖，使编写网页代码更加简单轻松。根据所获取页面中元素的不同，可以把 CSS3 选择器分为五大类：元素选择器、关系选择器、伪类选择器、伪对象选择器和属性选择器。本章将详细介绍 CSS3 各类选择器的使用。

【学习重点】

⏭ 正确使用 CSS 基本选择器。

⏭ 灵活使用组合选择器。

⏭ 掌握属性选择器和伪类选择器的应用。





2.1　CSS3 选择器概述

根据选择器结构的不同，可以把 CSS 选择器分为五大类。

☑　元素选择器，如表 2.1 所示。

表 2.1　元素选择器列表

选　择　器	说　　明
*	通配选择器，选定所有对象
E	标签选择器，匹配所有同类标签的元素
.className	类选择器，匹配 class 属性值包含 className 的元素。注意，E.className 表示限定元素类选择器
#IDName	ID 选择器，匹配 id 属性值等于 IDName 的元素。注意，E#IDName 表示限定元素 ID 选择器

☑　关系选择器，如表 2.2 所示。

表 2.2　关系选择器列表

选　择　器	说　　明
E,F	分组选择器，同时匹配 E 和 F 两个子选择器匹配的对象，子选择器之间用逗号分隔
E F	包含选择器，匹配所有被 E 元素包含的 F 元素
E > F	子选择器，匹配 E 元素的所有子元素 F
E + F	相邻选择器，匹配紧贴在 E 元素之后的 F 元素，元素 E 与 F 必须同属一个父级
E ~ F	兄弟选择器，匹配 E 元素后面的所有兄弟元素 F，元素 E 与 F 必须同属一个父级（**CSS3 新增**）

☑　属性选择器，如表 2.3 所示。

表 2.3　属性选择器列表

选　择　器	说　　明
E[att]	匹配具有 att 属性的 E 元素。注意，E 可以省略，如[cheacked]，以下相同
E[att="val"]	匹配具有 att 属性且属性值等于 val 的 E 元素
E[att~="val"]	匹配具有 att 属性且属性值为一用空格分隔的字词列表，其中一个等于 val 的 E 元素。注意，包含只有一个值且该值等于 val 的情况
E[att|="val"]	匹配具有 att 属性且其值是以 val 开头并用连接符“-”分隔的字符串的 E 元素。注意，如果值仅为 val，也将被选择
E[att^="val"]	匹配具有 att 属性且属性值为以 val 开头的字符串的 E 元素（**CSS3 新增**）
E[att$="val"]	匹配具有 att 属性且属性值为以 val 结尾的字符串的 E 元素（**CSS3 新增**）
E[att*="val"]	匹配具有 att 属性且属性值为包含 val 的字符串的 E 元素（**CSS3 新增**）

☑　伪选择器，包括伪类选择器（如表 2.4 所示）和伪对象选择器（如表 2.5 所示）。根据执行任务不同，伪类选择器又分为动态伪类、目标伪类、语言伪类、状态伪类、结构伪类和否定伪类 6 种。





表 2.4　伪类选择器列表

选　择　器	说　　明
E:link	设置超链接 a 在未被访问时的样式
E:visited	设置超链接 a 在其链接地址已被访问时的样式
E:hover	设置元素在其鼠标悬停时的样式
E:active	设置元素在被用户激活（在鼠标单击与释放之间发生的事件）时的样式
E:focus	设置对象在成为输入焦点时的样式
E:lang(fr)	匹配使用特殊语言的 E 元素
E:not(s)	匹配不含有 s 选择符的元素 E **（CSS3 新增）**
E:root	匹配 E 元素在文档的根元素。在 HTML 中，根元素永远是 HTML **（CSS3 新增）**
E:first-child	匹配父元素的第一个子元素 E **（CSS3 新增）**
E:last-child	匹配父元素的最后一个子元素 E **（CSS3 新增）**
E:only-child	匹配父元素仅有的一个子元素 E **（CSS3 新增）**
E:nth-child(n)	匹配父元素的第 n 个子元素 E，假设该子元素不是 E，则选择符无效 **（CSS3 新增）**
E:nth-last-child(n)	匹配父元素的倒数第 n 个子元素 E，假设该子元素不是 E，则选择符无效 **（CSS3 新增）**
E:first-of-type	匹配同类型中的第一个同级兄弟元素 E **（CSS3 新增）**
E:last-of-type	匹配同类型中的最后一个同级兄弟元素 E **（CSS3 新增）**
E:only-of-type	匹配同类型中的唯一的一个同级兄弟元素 E **（CSS3 新增）**
E:nth-of-type(n)	匹配同类型中的第 n 个同级兄弟元素 E **（CSS3 新增）**
E:nth-last-of-type(n)	匹配同类型中的倒数第 n 个同级兄弟元素 E **（CSS3 新增）**
E:empty	匹配没有任何子元素（包括 text 节点）的元素 E **（CSS3 新增）**
E:checked	匹配用户界面处于选中状态的元素 E。注意，用于 input 的 type 为 radio 与 checkbox 时 **（CSS3 新增）**
E:enabled	匹配用户界面上处于可用状态的元素 E **（CSS3 新增）**
E:disabled	匹配用户界面上处于禁用状态的元素 E **（CSS3 新增）**
E:target	匹配相关 URL 指向的 E 元素 **（CSS3 新增）**
@page :first	设置在打印时页面容器第一页使用的样式。注意，仅用于@page 规则
@page :left	设置页面容器位于装订线左边的所有页面使用的样式。注意，仅用于@page 规则
@page :right	设置页面容器位于装订线右边的所有页面使用的样式。注意，仅用于@page 规则

表 2.5　伪对象选择器列表

选　择　器	说　　明
E:first-letter/E::first-letter	设置对象内的第一个字符的样式。注意，仅作用于块对象 **（CSS3 新增）**
E:first-line/E::first-line	设置对象内的第一行的样式。注意，仅作用于块对象 **（CSS3 新增）**
E:before/E::before	设置在对象前发生的内容。与 content 属性一起使用，且必须定义 content 属性 **（CSS3 新增）**
E:after/E::after	设置在对象后发生的内容。与 content 属性一起使用，且必须定义 content 属性 **（CSS3 新增）**
E::placeholder	设置对象文字占位符的样式 **（CSS3 新增）**
E::selection	设置对象被选择时的样式 **（CSS3 新增）**



选择器模块权威参考：http://www.w3.org/TR/css3-selectors/。



提示：CSS 支持并列使用多个属性选择器，以匹配同时满足多个选择器，如 blockquote [class= quote][cite] {color:#f00;}。



注意：CSS3 将伪对象选择符前面的单个冒号（:）修改为双冒号（::），用以区别伪类选择符，但以前的写法仍然有效。

2.2　元素选择器

元素选择器包括标签选择器、类选择器、ID 选择器和通配选择器。



2.2.1　标签选择器

标签选择器也称为类型选择器，它直接引用 HTML 标签名称，用来匹配同名的所有标签。

☑　优点：使用简单，直接引用，不需要为标签添加属性。

☑　缺点：匹配的范围过大，精度不够。

因此，一般常用标签选择器重置各个标签的默认样式。

【示例】下面示例统一定义网页中段落文本的样式为：段落内文本字体大小为 12px，字体颜色为红色。实现该效果，可以考虑选用标签选择器定义如下样式。

```
p {
    font-size:12px;                    /*字体大小为 12px*/
    color:red;                         /*字体颜色为红色*/
}
```



2.2.2　类选择器

类选择器以点号（.）为前缀，后面是一个类名。应用方法：在标签中定义 class 属性，然后设置属性值为类选择器的名称。

☑　优点：能够为不同标签定义相同样式；使用灵活，可以为同一个标签定义多个类样式。

☑　缺点：需要为标签定义 class 属性，影响文档结构，操作相对麻烦。

【示例】下面示例演示如何在对象中应用多个样式类。

【操作步骤】

第 1 步，新建 HTML5 文档，保存为 test.html。

第 2 步，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表。

第 3 步，在<style>标签内输入下面样式代码，定义 3 个类样式：red、underline 和 italic。

```
/*颜色类*/
.red {color: red;}                            /*红色*/
/*下画线类*/
.underline {text-decoration: underline;}      /*下画线*/
/*斜体类*/
.italic {font-style: italic;}
```



第 4 步，在段落文本中分别引用这些类，其中第 2 段文本标签引用了 3 个类，演示效果如图 2.1 所示。



Note

```
<p class="underline">问君能有几多愁，恰似一江春水向东流。</p>
<p class="red italic underline">剪不断，理还乱，是离愁。别是一般滋味在心头。</p>
<p class="italic">独自莫凭栏，无限江山，别时容易见时难。流水落花春去也，天上人间。</p>
```

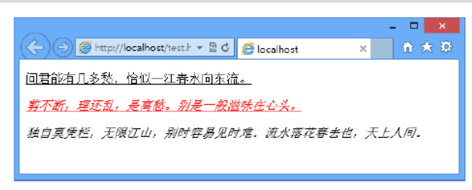


图 2.1　多类应用效果

2.2.3　ID 选择器



视频讲解

ID 选择器以井号（#）为前缀，后面是一个 ID 名。应用方法：在标签中定义 id 属性，然后设置属性值为 ID 选择器的名称。

☑　优点：精准匹配。

☑　缺点：需要为标签定义 id 属性，影响文档结构，相对于类选择器，缺乏灵活性。

【示例】下面示例演示如何在文档中应用 ID 选择器。

【操作步骤】

第 1 步，启动 Dreamweaver，新建一个网页，在<body>标签内输入<div>标签。

```
<div id="box">问君能有几多愁，恰似一江春水向东流。</div>
```

第 2 步，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表。

第 3 步，输入下面样式代码，为该盒子定义固定宽和高，并设置背景图像，以及边框和内边距大小。

```
#box {/*ID 样式*/
    background:url(images/1.png) center bottom;    /*定义背景图像并让其居中、底部对齐*/
    height:200px;                                  /*固定盒子的高度*/
    width:400px;                                   /*固定盒子的宽度*/
    border:solid 2px red;                          /*边框样式*/
    padding:100px;                                 /*增加内边距*/
}
```

第 4 步，在浏览器中预览，效果如图 2.2 所示。



图 2.2　ID 选择器的应用



> 提示：不管是类选择器，还是 ID 选择器，都可以指定一个限定标签名，用于限定它们的应用范围。例如，针对上面示例，在 ID 选择器前面增加一个<div>标签，这样 div#box 选择器的优先级会高于#box 选择器的优先级。在同等条件下，浏览器会优先解析 div#box 选择器定义的样式。对于类选择器，也可以使用这种方式限制其应用范围，并增加其优先级。



Note

2.2.4　通配选择器



视频讲解

通配选择器使用星号（*）表示，用来匹配文档中所有标签。

【示例】使用下面样式可以清除所有标签的边距。

```
* {margin: 0; padding: 0;}
```

2.3　关系选择器

当把两个简单的选择器组合在一起，就形成了一个复杂的关系选择器，通过关系选择器可以精确匹配 HTML 结构中特定范围的元素。

2.3.1　包含选择器



视频讲解

包含选择器通过空格连接两个简单的选择器，前面选择器表示包含的对象，后面选择器表示被包含的对象。

☑　优点：可以缩小匹配范围。

☑　缺点：匹配范围相对较大，影响的层级不受限制。

【示例】启动 Dreamweaver，新建一个网页，在<body>标签内输入如下结构：

```
<div id="wrap">
    <div id="header">
        <p>头部区域段落文本</p>
    </div>
    <div id="main">
        <p>主体区域段落文本</p>
    </div>
</div>
```

在<head>标签内添加<style type="text/css">标签，定义一个内部样式表。然后定义样式，希望实现如下设计目标。

☑　定义<div id="header">包含框内的段落文本字体大小为 14px。

☑　定义<div id="main">包含框内的段落文本字体大小为 12px。

这时可以利用包含选择器来快速定义样式，代码如下：

```
#header p {font-size:14px;}
#main p {font-size:12px;}
```

2.3.2　子选择器



视频讲解

子选择器使用尖角号（>）连接两个简单的选择器，前面选择器表示包含的父对象，后面选择器



表示被包含的子对象。

☑ 优点：相对于包含选择器，匹配的范围更小，从层级结构上看，匹配目标更明确。

☑ 缺点：相对于包含选择器，匹配范围有限，需要熟悉文档结构。



【示例】新建网页，在<body>标签内输入如下结构：

```
<h2><span>虞美人・春花秋月何时了</span></h2>
<div><span>春花秋月何时了？往事知多少。小楼昨夜又东风，故国不堪回首月明中。雕栏玉砌应犹在，只是朱颜改。问君能有几多愁？恰似一江春水向东流。 </span></div>
```

在<head>标签内添加<style type="text/css">标签，在内部样式表中定义所有 span 元素的字体大小为 18px，再用子选择器定义 h2 元素包含的 span 子元素的字体大小为 28px。

```
span {font-size: 18px;}
h2 > span {font-size: 28px;}
```

在浏览器中预览，显示效果如图 2.3 所示。

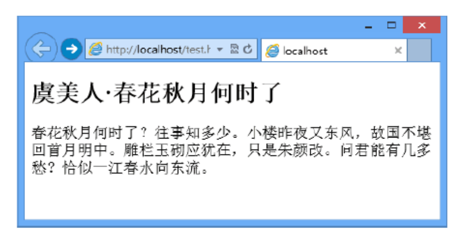


图 2.3 子选择器应用



视 频 讲 解

2.3.3 相邻选择器

相邻选择器使用加号（+）连接两个简单的选择器，前面选择器指定相邻的前面一个元素，后面选择器指定相邻的后面一个元素。

☑ 优点：在结构中能够快速、准确地找到同级、相邻元素。

☑ 缺点：使用前需要熟悉文档结构。

【示例】下面示例通过相邻选择器快速匹配出标题下面相邻的 p 元素，并设计其包含的文本居中显示，效果如图 2.4 所示。

```
<style type="text/css">
h2, h2 + p {text-align: center;}
</style>
<h2>虞美人・春花秋月何时了</h2>
<p>李煜 </p>
<p>春花秋月何时了？往事知多少。小楼昨夜又东风，故国不堪回首月明中。 </p>
<p>雕栏玉砌应犹在，只是朱颜改。问君能有几多愁？恰似一江春水向东流。 </p>
```



图 2.4 相邻选择器的应用







如果不使用相邻选择器，需要使用类选择器来设计，这样就相对麻烦很多。

2.3.4 兄弟选择器

兄弟选择器使用波浪符号（~）连接两个简单的选择器，前面选择器指定同级的前置元素，后面选择器指定其后同级所有匹配的元素。

☑ 优点：在结构中能够快速、准确地找到同级靠后的元素。

☑ 缺点：使用前需要熟悉文档结构，匹配精度没有相邻选择器具体。

【示例】以 2.3.3 节示例为基础，添加如下样式，定义标题后面所有段落文本的字体大小为 14px，字体颜色为红色。

```
h2 ~ p {font-size: 14px; color:red;}
```

在浏览器中预览，页面效果如图 2.5 所示。可以看到兄弟选择器匹配的范围包含了相邻选择器匹配的元素。

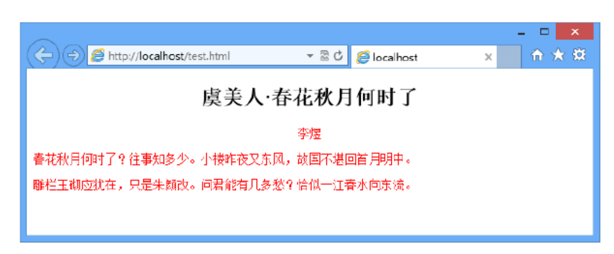


图 2.5　兄弟选择器的应用

2.3.5 分组选择器



分组选择器使用逗号（,）连接两个简单的选择器，前面选择器匹配的元素与后面选择器匹配的元素混合在一起作为分组选择器的结果集。

☑ 优点：可以合并相同样式，减少代码冗余。

☑ 缺点：不方便个性管理和编辑。

【示例】下面示例使用分组选择器将所有标题元素统一样式。

```
h1, h2, h3, h4, h5, h5, h6 {
    margin: 0;                    /*清除标题的默认外边距*/
    margin-bottom: 10px;          /*使用下边距拉开标题距离*/
}
```

2.4 属性选择器



属性选择器是根据标签的属性来匹配元素，使用中括号进行标识：

```
[属性表达式]
```

CSS3 包括 7 种属性选择器形式，下面结合示例具体说明。

【示例】下面设计一个简单的图片灯箱导航示例。其中 HTML 结构如下：

```
<div class="pic_box">
    <img src="images/bg1.jpg" />
```



```
        <div class="nav">
            <a href="#1" class="links item first" title="w3cplus" target="_blank" id="first" >1</a>
            <a href="#2" class="links active item" title="test website" target="_blank" lang="zh">2</a>
            <a href="#3" class="links item" title="this is a link" lang="zh-cn">3</a>
            <a href="#4" class="links item" target="_balnk" lang="zh-tw">4</a>
            <a href="#5" class="links item" title="zh-cn">5</a>
            <a href="#6" class="links item" title="website link" lang="zh">6</a>
            <a href="#7" class="links item" title="open the website" lang="cn">7</a>
            <a href="#8" class="links item" title="close the website" lang="en-zh">8</a>
            <a href="#9" class="links item" title="http://www.baidu.com">9</a>
            <a href="#10" class="links item last" id="last">10</a>
        </div>
    </div>
```

使用 CSS 适当美化，具体样式代码请参考本节示例源代码，初始预览效果如图 2.6 所示。

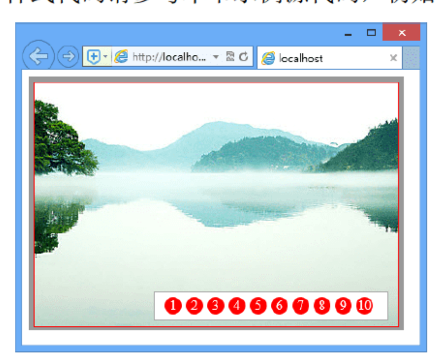


图 2.6　设计的灯箱广告效果图

1．E[attr]

选择具有 attr 属性的 E 元素。例如：

```
.nav a[id] {background: blue; color:yellow;font-weight:bold;}
```

上面代码表示选择 div.nav 下所有带有 id 属性的 a 元素，并在这个元素上使用背景色为蓝色，前景色为黄色，字体加粗的样式。对照上面的 HTML 结构，不难发现，只有第一个和最后一个链接使用了 id 属性，所以选中了这两个 a 元素，效果如图 2.7 所示。

也可以指定多属性：

```
.nav a[href][title] {background: yellow; color:green;}
```

上面代码表示选择 div.nav 下同时具有 href 和 title 两个属性的 a 元素，效果如图 2.8 所示。



图 2.7　属性快速匹配



图 2.8　多属性快速匹配

2．E[attr="value"]

选择具有 attr 属性且属性值等于 value 的 E 元素。例如：

```
.nav a[id="first"] {background: blue; color:yellow;font-weight:bold;}
```



上面代码表示选中 div.nav 中的 a 元素，这个元素有一个 id="first"属性值，预览效果如图 2.9 所示。

E[attr="value"]属性选择器也可以多个属性并写，进一步缩小选择范围，用法如下：

```
.nav a[href="#1"][title] {background: yellow; color:green;}
```

预览效果如图 2.10 所示。



图 2.9　属性值快速匹配



图 2.10　多属性值快速匹配

3．E[attr~="value"]

选择具有 attr 属性且属性值为一用空格分隔的字词列表，其中一个等于 value 的 E 元素。包含只有一个值且该值等于 val 的情况。例如：

```
.nav a[title~="website"]{background:orange;color:green;}
```

上面代码表示在 div.nav 下的 a 元素的 title 属性中，只要其属性值中含有 website 这个词就会被选择，结果“2”“6”“7”“8”这 4 个 a 元素的 title 中都含有，所以被选中，如图 2.11 所示。

4．E[attr^="value"]

选择具有 attr 属性且属性值为以 value 开头的字符串的 E 元素。例如：

```
.nav a[title^="http://"]{background:orange;color:green;}
.nav a[title^="mailto:"]{background:green;color:orange;}
```

上面代码表示选择含有 title 属性，并且属性值以 http://和 mailto:开头的所有 a 元素，匹配效果如图 2.12 所示。



图 2.11　属性值局部词匹配



图 2.12　匹配属性值开头字符串的元素

5．E[attr$="value"]

选择具有 attr 属性且属性值为以 value 结尾的字符串的 E 元素。例如：

```
.nav a[href$="png"]{background:orange;color:green;}
```

上面代码表示选择 div.nav 中元素有 href 属性且属性值以 png 结尾的 a 元素。

6．E[attr*="value"]

选择具有 attr 属性且属性值为包含 value 的字符串的 E 元素。例如：

```
.nav a[title*="site"]{background:black;color:white;}
```

上面代码表示选择 div.nav 中 title 属性值中有 site 字符串的 a 元素。上面样式的预览效果如图 2.13 所示。

7．E[attr|="value"]

选择具有 attr 属性且其值是以 value 开头，并用连接符“-”分隔的字符串的 E 元素；如果值仅为 value，也将被选择。例如：

```
.nav a[lang|="zh"]{background:gray;color:yellow;}
```



上面代码会选中 div.nav 中 lang 属性等于 zh 或以 zh-开头的所有 a 元素，如图 2.14 所示。



图 2.13　匹配属性值中的特定子串



图 2.14　匹配属性值开头字符串的元素



2.5　伪类选择器

伪类选择器是一种特殊的类选择器，它的用处就是可以对不同状态或行为下的元素定义样式，这些状态或行为是无法通过静态的选择器匹配的，具有动态特性。

2.5.1　伪选择器概述

伪选择器包括伪类选择器和伪对象选择器，伪选择器能够根据元素或对象的特征、状态、行为进行匹配。

伪选择器以冒号（:）作为前缀标识符。冒号前可以添加限定选择符，限定伪类应用的范围，冒号后为伪类和伪对象名，冒号前后没有空格。

CSS 伪类选择器有两种用法。

1．单纯式

```
E:pseudo-class {property:value}
```

其中 E 为元素，pseudo-class 为伪类名称，property 为 CSS 的属性，value 为 CSS 的属性值。例如：

```
a:link {color:red;}
```

2．混用式

```
E.class:pseudo-class{property:value}
```

其中.class 表示类选择符。把类选择符与伪类选择符组成一个混合式的选择器，能够设计更复杂的样式，以精准匹配元素。例如：

```
a.selected:hover {color: blue;}
```

CSS3 支持的伪类选择器具体说明如表 2.4 所示，CSS3 支持的伪对象选择器具体说明如表 2.5 所示。

由于 CSS3 伪选择器众多，下面仅针对 CSS3 中新增的伪类选择器进行说明，其他选择器请读者参考 CSS3 参考手册详细了解。



视 频 讲 解

2.5.2　结构伪类

结构伪类是根据文档结构的相互关系来匹配特定的元素，从而减少文档元素的 class 属性和 ID 属性的无序设置，使得文档更加简洁。

结构伪类形式多样，但用法固定，以便设计各种特殊样式效果。结构伪类主要包括下面几种。

☑　:first-child：第一个子元素。
☑　:last-child：最后一个子元素。





- ☑ :nth-child()：按正序匹配特定子元素。
- ☑ :nth-last-child()：按倒序匹配特定子元素。
- ☑ :nth-of-type()：在同类型中匹配特定子元素。
- ☑ :nth-last-of-type()：按倒序在同类型中匹配特定子元素。
- ☑ :first-of-type：第一个同类型子元素。
- ☑ :last-of-type：最后一个同类型子元素。
- ☑ :only-child：唯一子元素。
- ☑ :only-of-type：同类型的唯一子元素。
- ☑ :empty：空元素。

【示例 1】下面示例设计排行榜栏目列表样式，设计效果如图 2.15 所示。在列表框中为每个列表项定义相同的背景图像。

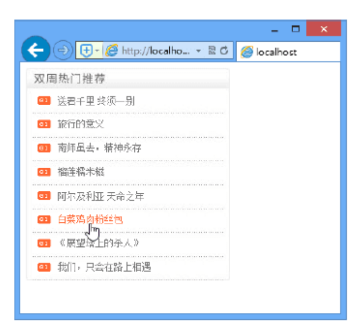


图 2.15　排行榜栏目样式

设计列表结构如下：

```
<div id="wrap">
    <ul id="container">
        <li><a href="#">送君千里 终须一别</a></li>
        <li><a href="#">旅行的意义</a></li>
        <li><a href="#">南师虽去，精神永存</a></li>
        <li><a href="#">榴莲糯米糍</a></li>
        <li><a href="#">阿尔及利亚 天命之年</a></li>
        <li><a href="#">白菜鸡肉粉丝包</a></li>
        <li><a href="#">《展望塔上的杀人》</a></li>
        <li><a href="#">我们，只会在路上相遇</a></li>
    </ul>
</div>
```

设计的列表样式请参考本节示例源代码。下面以示例 1 为基础分析结构伪类选择器的用法。

1．:first-child

【示例 2】设计第一个列表项前的图标为 1，且字体加粗显示。

使用:first-child 匹配，代码如下：

```
#wrap li:first-child {
    background-position:2px 10px;
    font-weight:bold;
}
```

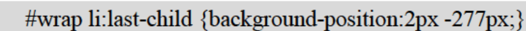


Note

2．: last-child

【示例 3】单独给最后一个列表项定义样式。

使用:last-child 来匹配，代码如下：



```
#wrap li:last-child {background-position:2px -277px;}
```

显示效果如图 2.16 所示。

3．:nth-child()

:nth-child()可以选择一个或多个特定的子元素。该函数有多种用法：

```
:nth-child(length);                    /*参数是具体数字*/
:nth-child(n);                         /*参数是 n，n 从 0 开始计算*/
:nth-child(n*length)                   /*n 的倍数选择，n 从 0 开始计算*/
:nth-child(n+length);                  /*选择大于或等于 length 的元素*/
:nth-child(-n+length)                  /*选择小于或等于 length 的元素*/
:nth-child(n*length+1);                /*表示隔几选一*/
```

在 nth-child()函数中，参数 length 和 n 均为整数。

:nth-child()可以定义值，值可以是整数，也可以是表达式，用来选择特定的子元素。

【示例 4】下面 6 个样式分别匹配列表中第 2～7 个列表项，并分别定义它们的背景图像 Y 轴坐标位置，显示效果如图 2.17 所示。

```
#wrap li:nth-child(2) {background-position: 2px -31px;}
#wrap li:nth-child(3) {background-position: 2px -72px;}
#wrap li:nth-child(4) {background-position: 2px -113px;}
#wrap li:nth-child(5) {background-position: 2px -154px;}
#wrap li:nth-child(6) {background-position: 2px -195px;}
#wrap li:nth-child(7) {background-position: 2px -236px;}
```



图 2.16　设计最后一个列表项样式



图 2.17　设计每个列表项样式

注意：:nth-child()函数的参数不能引用负值，如 li:nth-child(-3)是不正确的使用方法。

（1）:nth-child(n)

在:nth-child(n)中，n 是一个简单的表达式，其取值从 0 开始计算，到什么时候结束是不确定的，须结合文档结构而定，如果在实际应用中直接这样使用，将会选中所有子元素。

在示例 4 中，如果在 li 中使用:nth-child(n)，那么将选中所有的 li 元素：

```
#wrap li:nth-child(n) {text-decoration:underline;}
```



这个样式类似于：

```
#wrap li {text-decoration:underline;}
```

其实，nth-child()是这样计算的：

n=0 表示没有选择元素。

n=1 表示选择第 1 个 li。

n=2 表示选择第 2 个 li。

依此类推，这样下来就选中了所有的 li。

（2）:nth-child(2n)

:nth-child(2n)是:nth-child(n)的一种变体，使用它可以选择 n 的倍数（其中 2 可以换成需要的数字，分别表示不同的倍数）。

```
#wrap li:nth-child(2n) {font-weight:bold;}
```

等价于：

```
#wrap li:nth-child(even) {font-weight:bold;}
```

预览效果如图 2.18 所示。



图 2.18　设计偶数行列表项样式

其实现过程如下：

当 n=0，则 2n=0，表示没有选中任何元素。

当 n=1，则 2n=2，表示选择第 2 个 li。

当 n=2，则 2n＝4，表示选择第 4 个 li。

依此类推。

如果是 2n，则与使用 even 命名 class 定义样式所起到的效果是一样的。

（3）:nth-child(2n-1)

:nth-child(2n-1)选择器是在:nth-child(2n)基础上演变来的，既然:nth-child(2n)表示选择偶数，那么在它的基础上减去 1 就变成选择奇数。

```
#wrap li:nth-child(2n-1) {font-weight:bold;}
```

等价于：

```
#wrap li:nth-child(odd) {font-weight:bold;}
```

其实现过程如下：

当 n=0，则 2n-1=-1，表示没有选中任何元素。

当 n=1，则 2n-1=1，表示选择第 1 个 li。

当 n=2，则 2n-1=3，表示选择第 3 个 li。

依此类推。

奇数效果还可以使用:nth-child(2n+1)和:nth-child(odd)来实现。

（4）:nth-child(n+5)

:nth-child(n+5)选择器是从第 5 个子元素开始选择。

```
li:nth-child(n+5) {font-weight:bold;}
```

其实现过程如下：

当 n=0，则 n+5=5，表示选择第 5 个 li。

Note





Note

当 n=1，则 n+5=6，表示选择第 6 个 li。

依此类推。

可以使用这种方法设置需要开始选择的元素位置，也就是说更改数字，可更改起始位置。

（5）:nth-child(-n+5)

:nth-child(-n+5)选择器刚好和:nth-child(n+5)选择器相反，它选择第 5 个元素前面的子元素。

```
li:nth-child(-n+5) {font-weight:bold;}
```

其实现过程如下：

当 n=0，则-n+5=5，表示选择第 5 个 li。

当 n=1，则-n+5=4，表示选择第 4 个 li。

当 n=2，则-n+5=3，表示选择第 3 个 li。

当 n=3，则-n+5=2，表示选择第 2 个 li。

当 n=4，则-n+5=1，表示选择第 1 个 li。

当 n=5，则-n+5=0，表示没有选择任何元素。

（6）:nth-child(5n+1)

:nth-child(5n+1)选择器是实现隔几选一的效果。如果是隔三选一，则定义的样式如下：

```
li:nth-child(3n+1) {font-weight:bold;}
```

其实现过程如下：

当 n=0，则 3n+1=1，表示选择第 1 个 li。

当 n=1，则 3n+1=4，表示选择第 4 个 li。

当 n=2，则 3n+1=7，表示选择第 7 个 li。

设计效果如图 2.19 所示。



图 2.19　设计隔三选一行列表项样式

4．:nth- last-child()

:nth-last-child()选择器与:nth-child()相似，但作用与:nth-child 不一样，:nth-last-child()是从最后一个元素开始计算，来选择特定元素。

```
li:nth-last-child(4) {font-weight:bold;}
```

上面代码表示选择倒数第 4 个列表项。

其中:nth-last-child(1)和:last-child 所起作用是一样的，都表示选择最后一个元素。

另外，:nth-last-child()与:nth-child()用法相同，可以使用表达式来选择特定元素，下面来看几个特殊的表达式所起的作用。



:nth-last-child(2n)表示从元素后面计算，选择的是偶数个数，而从前面来说就是选择元素的奇数个数，与:nth-child(2n+1)、:nth-child(2n-1)、:nth-child(odd)所起的作用是一样的。如：

```
li:nth-last-child(2n) {font-weight:bold;}
li:nth-last-child(even) {font-weight:bold;}
```

等价于：

```
li:nth-child(2n+1) {font-weight:bold;}
li:nth-child(2n-1) {font-weight:bold;}
li:nth-child(odd) {font-weight:bold;}
```

:nth-last-child(2n-1)选择器刚好与:nth-last-child(2n)相反，它从后面计算选择的是奇数，而从前面计算选择的就是偶数位了，如：

```
li:nth-last-child(2n+1) {font-weight:bold;}
li:nth-last-child(2n-1) {font-weight:bold;}
li:nth-last-child(odd) {font-weight:bold;}
```

等价于：

```
li:nth-child(2n) {font-weight:bold;}
li:nth-child(even) {font-weight:bold;}
```

总之，:nth-last-child()和 nth-child()使用方法是一样的，只不过:nth-child()是从元素的第一个开始计算，而:nth-last-child()是从元素的最后一个开始计算，它们的计算方法都是一样的。

5．:nth-of-type()

:nth-of-type()类似于:nth-child()，不同的是它只计算选择器中指定的那个元素。:nth-of-type()选择器用来定位元素中包含好多不同类型的子元素时很有用处。

【示例 5】在 div#wrap 中有很多 p、li、img 等元素，但现在只需要选择 p 元素，并让它每隔一个 p 元素就有不同的样式，那就可以简单地写成：

```
div#wrap p:nth-of-type(even) {font-weight:bold;}
```

其实，这种用法与:nth-child()是一样的，也可以使用:nth-child()的表达式来实现，唯一不同的是:nth-of-type 指定了元素的类型。

6．:nth-last-of-type()

:nth-last-of-type()与:nth-last-child()用法相同，但它指定了子元素的类型。除此之外，语法形式和用法基本相同。

7．:first-of-type 和:last-of-type

:first-of-type 和:last-of-type 选择器类似于:first-child 和:last-child，不同之处就是它们指定了元素的类型。

8．:only-child 和:only-of-type

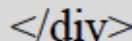
:only-child 表示一个元素是它的父元素的唯一子元素。

【示例 6】在文档中设计如下 HTML 结构。

```
<div class="post">
    <p>第一段文本内容</p>
```



```
        <p>第二段文本内容</p>
    </div>
    <div class="post">
        <p>第三段文本内容</p>
    </div>
```



如果需要使 div.post 只有一个 p 元素时，改变 p 的样式，可以使用:only-child 选择器来实现。

```
.post p {font-weight:bold;}
.post p:only-child {background: red;}
```

此时若 div.post 只有一个子元素 p，那么它的背景色将会显示为红色。

:only-of-type 表示一个元素包含很多个子元素，而其中只有一个子元素是唯一的，那么使用这种选择方法就可以选中这个唯一的子元素。例如：

```
<div class="post">
    <div>子块一</div>
    <p>文本段</p>
    <div>子块二</div>
</div>
```

如果只想选择上面结构块中的 p 元素，就可以这样写：

```
.post p:only-of-type{background-color:red;}
```

9．:empty

:empty 用来选择没有任何内容的元素，这里没有内容指的是一点内容都没有，包括空格。

【示例 7】以下有 3 个段落，其中一个段落什么都没有，完全是空的：

```
<div class="post">
    <p>第一段文本内容</p>
    <p>第二段文本内容</p>
</div>
<div class="post">
    <p> </p>
</div>
```

如果想设计这个 p 不显示，那就可以这样来写：

```
.post p:empty {display: none;}
```


视频讲解

2.5.3 否定伪类

:not()表示否定选择器，即过滤掉 not()函数匹配的特定元素。

【示例】下面示例为页面中所有段落文本设置字体大小为 24px，然后使用:not(.author)排除第一段文本，设置其他段落文本的字体大小为 14px，显示效果如图 2.20 所示。

```
<style type="text/css">
p {font-size: 24px;}
p:not(.author){font-size: 14px;}
</style>
<h2>虞美人·春花秋月何时了</h2>
```



```
<p class="author">李煜 </p>
<p>春花秋月何时了？往事知多少。小楼昨夜又东风，故国不堪回首月明中。 </p>
<p>雕栏玉砌应犹在，只是朱颜改。问君能有几多愁？恰似一江春水向东流。 </p>
```



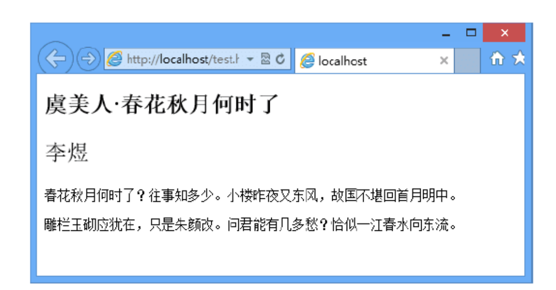


图 2.20　否定伪类的应用

2.5.4　状态伪类



CSS3 包含 3 个 UI 状态伪类选择器，简单说明如下。

☑　: enabled：匹配指定范围内所有可用 UI 元素。

☑　:disabled：匹配指定范围内所有不可用 UI 元素。

☑　:checked：匹配指定范围内所有可用 UI 元素。

【示例】下面示例设计一个简单的登录表单，效果如图 2.21 所示。在实际应用中，当用户登录完毕，不妨通过脚本把文本框设置为不可用（disabled="disabled"）状态，这时可以通过:disabled 选择器让文本框显示为灰色，以告诉用户该文本框已不可用，这样就不用设计“不可用”样式类，并把该类添加到 HTML 结构中。

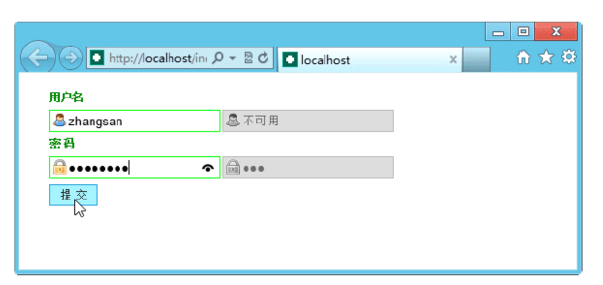


图 2.21　设计登录表单样式

【操作步骤】

第 1 步，新建一个文档，在文档中构建一个简单的登录表单结构。

```
<form action="#">
    <label for="username">用户名</label>
    <input type="text" name="username" id="username" />
    <input type="text" name="username1" disabled="disabled" value="不可用" />
    <label for="password">密 码</label>
    <input type="password" name="password" id="password" />
    <input type="password" name="password1" disabled="disabled" value="不可用" />
    <input type="submit" value="提 交" />
</form>
```



在这个表单结构中，使用 HTML 的 disabled 属性分别定义两个不可用的文本框对象。

第 2 步，内建一个内部样式表，使用属性选择器定义文本框和密码域的基本样式。



Note

```
input[type="text"], input[type="password"] {
    border:1px solid #0f0;
    width:160px;
    height:22px;
    padding-left:20px;
    margin:6px 0;
    line-height:20px;}
```

第 3 步，利用属性选择器，分别为文本框和密码域定义内嵌标识图标。

```
input[type="text"] {background:url(images/name.gif) no-repeat 2px 2px;}
input[type="password"] {background:url(images/password.gif) no-repeat 2px 2px;}
```

第 4 步，使用状态伪类选择器，定义不可用表单对象显示为灰色，以提示用户该表单对象不可用。

```
input[type="text"]:disabled {
    background:#ddd url(images/name1.gif) no-repeat 2px 2px;
    border:1px solid #bbb;}
input[type="password"]:disabled {
    background:#ddd url(images/password1.gif) no-repeat 2px 2px;
    border:1px solid #bbb;}
```



视 频 讲 解

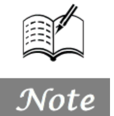
2.5.5 目标伪类

目标伪类选择器类型形式如 E:target，它表示选择匹配 E 的所有元素，且匹配元素被相关 URL 指向。该选择器是动态选择器，只有当存在 URL 指向该匹配元素时，样式效果才有效。

【示例】下面示例设计当单击页面中的锚点链接，跳转到指定标题位置时，该标题会自动高亮显示，以提醒用户当前跳转的位置，效果如图 2.22 所示。

```
<style type="text/css">
/*设计导航条固定在窗口右上角位置显示*/
h1{position:fixed; right:12px; top:24px;}
/*让锚点链接堆叠显示*/
h1 a{display:block;}
/*设计锚点链接的目标高亮显示*/
h2:target {background:hsla(93,96%,62%,1.00);}
</style>
<h1><a href="#p1">图片 1</a> <a href="#p2">图片 2</a> <a href="#p3">图片 3</a> <a href="#p4">图片 4</a>
</h1>
<h2 id="p1">图片 1</h2>
<p><img src="images/1.jpg" /></p>
<h2 id="p2">图片 2</h2>
<p><img src="images/2.jpg" /></p>
<h2 id="p3">图片 3</h2>
<p><img src="images/3.jpg" /></p>
<h2 id="p4">图片 4</h2>
<p><img src="images/4.jpg" /></p>
```





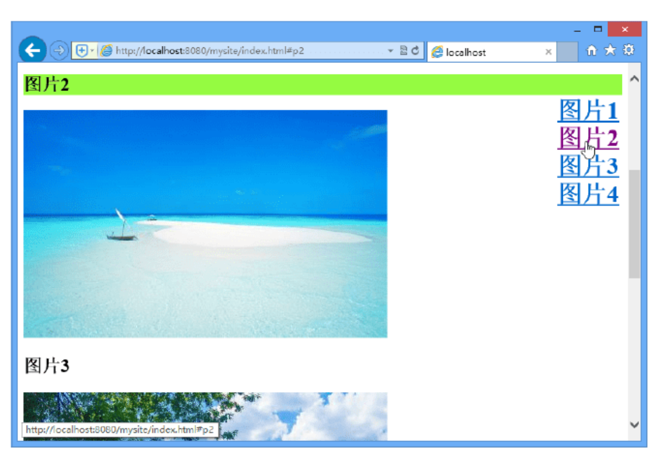


图 2.22　目标伪类样式应用效果

视 频 讲 解

2.5.6　动态伪类

动态伪类是一类行为类样式，只有当用户与页面进行交互时有效。包括以下两种形式。

☑　锚点伪类，如:link、:visited。

☑　行为伪类，如:hover、:active 和:focus。

【示例】下面示例使用动态伪类选择器设计一组 3D 动态效果的按钮样式，效果如图 2.23 所示。



图 2.23　设计 3D 按钮样式

【操作步骤】

第 1 步，设计一个 HTML 文档结构。创建一个新的 HTML 文档，并添加一个列表，在列表项中包含基本的锚链接。不需要任何额外的<div>或者<span>标签，也不用添加 id 和 class 属性，一切效果都通过 CSS 进行控制。结构代码如下：

```
<ul id="container">
    <li><a href="#" class="button gray">灰色风格按钮</a></li>
    <li><a href="#" class="button pink">粉红风格按钮</a></li>
    <li><a href="#" class="button blue">蓝色风格按钮</a></li>
    <li><a href="#" class="button green">绿色风格按钮</a></li>
    <li><a href="#" class="button turquoise">天蓝色风格按钮</a></li>
    <li><a href="#" class="button black">黑色风格按钮</a></li>
    <li><a href="#" class="button darkgray">深灰色风格按钮</a></li>
    <li><a href="#" class="button yellow">黄色风格按钮</a></li>
    <li><a href="#" class="button purple">紫色风格按钮</a></li>
    <li><a href="#" class="button darkblue">银灰风格按钮</a></li>
</ul>
```





为了演示不同色彩风格的按钮样式，可以定义主题样式类，通过给每一个按钮应用不同的主题样式，可以充分发挥CSS的优势和便捷之处。

第2步，新建内部样式表，定义基本的按钮类样式。

```
ul {list-style: none;}
a.button {
    display: block; float: left;
    position: relative;
    height: 25px; width: 120px;
    margin: 0 10px 18px 0;
    text-decoration: none;
    font: 12px "Helvetica Neue", Helvetica, Arial, sans-serif;
    font-weight: bold; line-height: 25px; text-align: center;
}
```

第3步，为按钮类样式增加行为样式，让按钮实现动态效果。这里主要使用了:hover伪类选择器。例如，为灰色系按钮设计鼠标经过时的动态样式效果，主要包括字体颜色和背景色的变化，以及边框线的变换，以模拟立体效果。

```
.gray, .gray:hover {
    color: #555;
    border-bottom: 4px solid #b2b1b1;
    background: #eee;
}
.gray:hover {background: #e2e2e2;}
```

第4步，定义双边框样式。通过预览会发现现在的按钮边框显得比较单薄，需要为按钮定义粗边框的底部效果，同时还需要增加一点点行间距，因此这里使用了:before和:after伪类样式。

```
a.button:before, a.button:after {
    content: '';
    position: absolute;
    left: -1px; bottom: -1px;
    height: 25px; width: 120px;
    border-radius: 3px;
}
a.button:before {
    height: 23px;
    bottom: -4px;
    border-top: 0;
    border-radius: 0 0 3px 3px;
    box-shadow: 0 1px 1px 0px #bfbfbf;
}
```

第5步，为了彰显按钮的金属特质，不妨借助CSS3的特效定义圆角效果。

```
a.button {border-radius: 3px;}
```

第6步，为边框定义阴影效果。

```
a.button:before, a.button:after {border-radius: 3px;}
a.button:before {
    border-radius: 0 0 3px 3px;
```



```
    box-shadow: 0 1px 1px 0px #bfbfbf;
}
```

第 7 步，定义鼠标经过和访问过按钮伪类状态类样式，设计渐变背景色特效。

```
/*设计灰色主题风格*/
a.gray, a.gray:hover, a.gray:visited {
    color: #555;
    border-bottom: 4px solid #b2b1b1;
    text-shadow: 0px 1px 0px #fafafa;
    background: #eee;
    background: linear-gradient(to top,  #eee,  #e2e2e2);
    box-shadow: inset 1px 1px 0 #f5f5f5;
}
.gray:before, .gray:after {
    border: 1px solid #cbcbcb;
    border-bottom: 1px solid #a5a5a5;
}
.gray:hover {
    background: #e2e2e2;
    background: linear-gradient(to top,  #e2e2e2,  #eee);
}
```



第 8 步，利用:active 伪类选择器定义对象激活状态的样式效果。

```
/*激活状态样式*/
a.button:active {
    border: none;
    bottom: -4px;
    margin-bottom: 22px;
    box-shadow:  1px 1px 0 #fff, inset 0 1px 1px rgba(0, 0, 0, 0.3);
}
a.button:active:before,
a.button:active:after {
    border: none;
    box-shadow: none;
}
```

2.6　伪对象选择器



伪对象选择器主要针对不确定对象定义样式，如第一行文本、第一个字符、前面内容、后面内容。这些对象具体存在，但又无法具体确定，需要使用特定类型的选择器来匹配它们。

伪对象选择器以冒号（:）作为语法标识符。冒号前可以添加选择符，限定伪对象应用的范围，冒号后为伪对象名称，冒号前后没有空格。语法格式如下：

```
:伪对象名称
```

CSS3 新语法格式如下：

```
::伪对象名称
```





提示：伪对象前面包含两个冒号，主要是为了与伪类选择器进行语法区分。

【示例 1】下面示例使用:before 伪对象选择器在段落文本前面添加 3 个字符“柳永：”，然后使用:first-letter 伪对象选择器设置段落文本第一个字符放大显示，定义字体大小为 24px，效果如图 2.24 所示。

```
<style type="text/css">
p:before {content: '柳永：';}
p:first-letter {font-size:24px;}
</style>
<p>衣带渐宽终不悔，为伊消得人憔悴。</p>
```

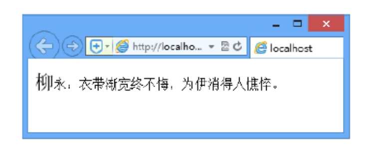


图 2.24　定义第一个字符放大显示

【示例 2】下面示例使用:first-letter 伪对象选择器设置段落文本第一个字符放大下沉显示，并使用:first-line 伪对象选择器设置段落文本第一行字符放大带有阴影显示，效果如图 2.25 所示。

```
<style type="text/css">
p{font-size:18px; line-height:1.6em;}
p:first-letter {/*段落文本中第一个字符样式*/
    float:left;
    font-size:60px;
    font-weight:bold;
    margin:26px 6px;
}
p:first-line {/*段落文本中第一行字符样式*/
    color:red;
    font-size:24px;
    text-shadow:2px 2px 2px rgba(147,251,64,1);
}
</style>

<p>我在朦胧中，眼前展开一片海边碧绿的沙地来，上面深蓝的天空中挂着一轮金黄的圆月。我想：希望本是无所谓有，无所谓无的。这正如地上的路；其实地上本没有路，走的人多了，也便成了路。 </p>
```

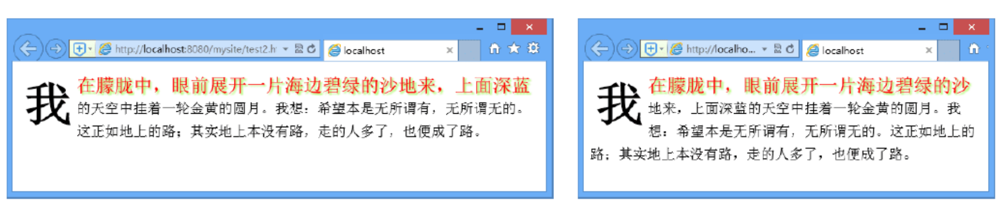


图 2.25　定义第一个字符和第一行字符特殊显示

2.7　案 例 实 战

下面以案例形式练习 CSS3 选择器的应用。





2.7.1　设计表格样式

本案例设计一个分层表格样式，借助否定伪类选择器和结构伪类选择器，配合 CSS 背景图像设计树形结构标志；借助伪类选择器设计鼠标经过时动态背景效果；利用 CSS 边框和背景色设计标题行的立体显示效果。演示效果如图 2.26 所示。

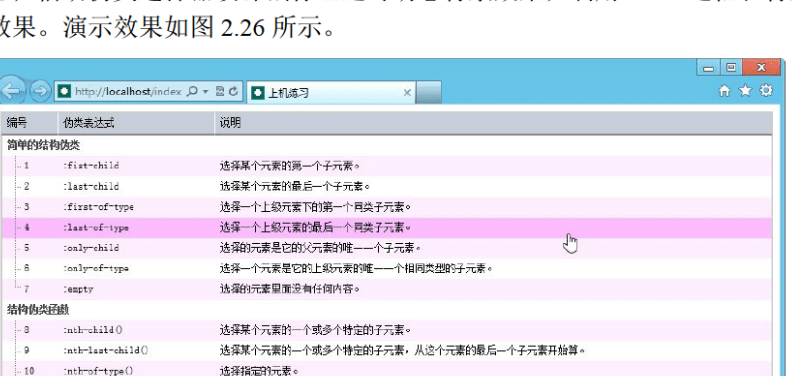


图 2.26　设计表格样式

【操作步骤】

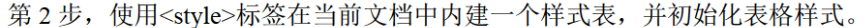
第 1 步，利用表格结构构建一个数据表。

```
<table>
    <thead>
        <tr>
            <th>编号</th>
            <th>伪类表达式</th>
            <th>说明</th>
        </tr>
    </thead>
    <tbody>
        <tr><td colspan="3">简单的结构伪类</td></tr>
        <tr><th>1</th><td>:first-child</td><td>选择某个元素的第一个子元素。</td></tr>
        <tr><th>2</th><td>:last-child</td><td>选择某个元素的最后一个子元素。</td></tr>
        <tr><th>3</th><td>:first-of-type</td><td>选择一个上级元素下的第一个同类子元素。</td></tr>
        <tr><th>4</th><td>:last-of-type</td><td>选择一个上级元素的最后一个同类子元素。</td></tr>
        <tr><th>5</th><td>:only-child</td><td>选择的元素是它的父元素的唯一一个子元素。</td></tr>
        <tr><th>6</th><td>:only-of-type</td><td>选择一个元素是它的上级元素的唯一一个相同类型的子
元素。</td></tr>
        <tr><th class="end">7</th><td>:empty</td><td>选择的元素里面没有任何内容。</td></tr>
        <tr><td colspan="3">结构伪类函数</td></tr>
        <tr><th>8</th><td>:nth-child()</td><td>选择某个元素的一个或多个特定的子元素。</td></tr>
        <tr><th>9</th><td>:nth-last-child()</td><td>选择某个元素的一个或多个特定的子元素，从这个元素
的最后一个子元素开始算。</td></tr>
        <tr><th>10</th><td>:nth-of-type()</td><td>选择指定的元素。</td></tr>
        <tr><th class="end">11</th><td>:nth-last-of-type()</td><td>选择指定的元素，从元素的最后一个开始
计算。</td></tr>
```



```
        </tbody>
    </table>
```

第 2 步，使用<style>标签在当前文档中内建一个样式表，并初始化表格样式。





Note

```
table {border-collapse: collapse; font-size: 75%; line-height: 1.4; border: solid 2px #ccc; width: 100%;}
th, td {padding: .3em .5em; cursor: pointer;}
th {font-weight: normal; text-align: left; padding-left: 15px;}
```

第 3 步，使用结构伪类选择器匹配合并单元格所在的行，定义合并单元格所在行加粗显示。

```
td:only-of-type {
    font-weight:bold;
    color:#444;
}
```

第 4 步，使用否定伪类选择器选择主体区域非最后一个 th 元素。以背景方式在行前定义结构路径线。

```
tbody th:not(.end) {
    background: url(images/dots.gif) 15px 56% no-repeat;
    padding-left: 26px;
}
```

第 5 步，使用类选择器选择主体区域非最后一个 th 元素。以背景方式在行前定义结构封闭路径线。

```
tbody th.end {
    background: url(images/dots3.gif) 15px 56% no-repeat;
    padding-left: 26px;
}
```

第 6 步，使用 thead 元素把表头标题独立出来，方便 CSS 控制，避免定义过多的 class 属性。th 元素有两种显示形式：一种用来定义列标题；另一种用来定义行标题。下面样式是定义表格标题列样式。

```
thead th {
    background: #c6ceda;
    border-color: #fff #fff #888 #fff;
    border-style: solid;
    border-width: 1px 1px 2px 1px;
    padding-left: .5em;
}
```

第 7 步，设计隔行换色的背景效果，这里主要应用了:nth-child(2n)选择器。同时使用:hover 动态伪类定义鼠标经过时的行背景色动画变化，以提示鼠标当前经过行效果。

```
tbody tr:nth-child(2n) {background-color: #fef;}
tbody tr:hover{background: #fbf;}
```



视 频 讲 解

2.7.2 设计超链接样式

本案例模拟百度文库的“相关文档推荐”模块样式设计效果，演示如何利用属性选择器快速并准确匹配文档类型，为不同类型文档超链接定义不同的显示图标，以便浏览者准确识别文档类型。示例演示效果如图 2.27 所示。





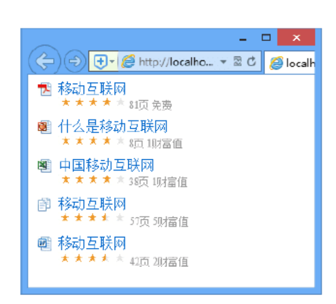


图 2.27　设计超链接文档类型的显示图标

【操作步骤】

第 1 步，构建一个简单的模块结构。在这个模块结构中，为了能够突出重点，忽略了其他细节信息。代码如下：

```
<div id="wrap">
    <p><a href="http://www.baidu.com/name.pdf">移动互联网</a><span><img src="images/star1.jpg" /> 81 页免费</span> </p>
    <p><a href="http://www.baidu.com/name.ppt">什么是移动互联网</a><span><img src="images/star1.jpg" /> 8 页  1 财富值</span> </p>
    <p><a href="http://www.baidu.com/name.xls">中国移动互联网</a><span><img src="images/star1.jpg" /> 38 页  1 财富值  </span> </p>
    <p><a href="http://www.baidu.com/name.txt">移动互联网</a> <span><img src="images/star3.jpg" /> 57 页  5 财富值</span></p>
    <p><a href="http://www.baidu.com/name.doc">移动互联网</a><span><img src="images/star3.jpg" /> 42 页  2 财富值</span> </p>
</div>
```

第 2 步，新建一个内部样式表，在样式表中对案例文档进行样式初始化，代码如下：

```
/*初始化超链接、span 元素和 p 元素基本样式*/
a {padding-left: 24px; text-decoration: none;}
span {color: #999; font-size: 12px; display: block; padding-left: 24px; padding-bottom: 6px;}
p {margin: 4px;}
```

第 3 步，利用属性选择器为不同类型文档超链接定义显示图标。

```
a[href$="pdf"] {/*匹配 PDF 文件*/
    background: url(images/pdf.jpg) no-repeat left center;
}
a[href$="ppt"] {/*匹配演示文稿*/
    background: url(images/ppt.jpg) no-repeat left center;
}
a[href$="txt"] {/*匹配记事本文件*/
    background: url(images/txt.jpg) no-repeat left center;
}
a[href$="doc"] {/*匹配 Word 文件*/
    background: url(images/doc.jpg) no-repeat left center;
}
a[href$="xls"] {/*匹配 Excel 文件*/
    background: url(images/xls.jpg) no-repeat left center;
}
```



【拓展】

超链接的类型和形式是多样的，如锚链接、下载链接、图片链接、空链接、脚本链接等，都可以利用属性选择器来标识这些超链接的不同样式。代码如下：



```
a[href^="http:"] {/*匹配所有有效超链接*/
    background: url(images/window.gif) no-repeat left center;
}
a[href$="xls"] {/*匹配 XML 样式表文件*/
    background: url(images/icon_xls.gif) no-repeat left center;
    padding-left: 18px;
}
a[href$="rar"] {/*匹配压缩文件*/
    background: url(images/icon_rar.gif) no-repeat left center;
    padding-left: 18px;
}
a[href$="gif"] {/*匹配 GIF 图像文件*/
    background: url(images/icon_img.gif) no-repeat left center;
    padding-left: 18px;
}
a[href$="jpg"] {/*匹配 JPG 图像文件*/
    background: url(images/icon_img.gif) no-repeat left center;
    padding-left: 18px;
}
a[href$="png"] {/*匹配 PNG 图像文件*/
    background: url(images/icon_img.gif) no-repeat left center;
    padding-left: 18px;
}
```

2.8 在线练习

本节为读者提供了多个课外练习，以便灵活使用 CSS，强化基本功训练。感兴趣的读者可以扫码练习。



在线练习



第3章 使用 CSS3 美化文本和图像

对网页进行修饰可以使其更加美观，CSS3 为字体和文本提供了大量的属性，如字体的类型、大小和颜色等，文本的对齐、间距、缩进和行高等。本章重点讲解 CSS3 字体和文本样式，由于文本与图片在网页中紧密排版在一起，本章还介绍了图片的常用样式设计。

【学习重点】

- 能够定义字体类型、大小、颜色等基本样式。
- 能够设计文本基本版式，如对齐、行高、间距等。
- 能够设计图片的基本样式。
- 能够灵活设计美观、实用的图文版式。





Note

3.1 设计字体样式

字体样式包括字体类型、大小、颜色、粗细、下画线、斜体、大小写等。下面分别进行介绍。



线上阅读



视频讲解

3.1.1 定义字体类型

使用 font-family 属性可以定义字体类型，用法如下：

```
font-family: name
```

其中 name 表示字体名称，可以设置字体列表，多个字体按优先顺序排列，以逗号隔开。

如果字体名称包含空格，则应使用引号括起。第二种声明方式使用所列出的字体序列名称，如果使用 fantasy 序列，将提供默认字体序列。

【示例】启动 Dreamweaver，新建一个网页，保存为 test1.html，在<body>标签内输入两行段落文本：

```
<p>月落乌啼霜满天，江枫渔火对愁眠。 </p>
<p>姑苏城外寒山寺，夜半钟声到客船。</p>
```

在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后输入下面样式，用来定义网页字体的类型。

```
p {/*段落样式*/
    font-family:   "隶书";                    /*隶书字体*/
}
```

在浏览器中预览效果如图 3.1 所示。



图 3.1　设计隶书字体效果

提示：在网页设计中，没有中文通用字体类型，中文字体的表现力比较弱，即使存在各种艺术字体，但是考虑到用户系统的支持率，很少被广泛使用。一般中文网页字体默认为宋体，对于标题或特殊提示信息，如果需要特殊字体，则建议采用图像形式间接实现。

拉丁字体类型比较丰富，通用字体的选择余地大、艺术表现力强，在浏览外文网站时，用户会发现页面选用的字体类丰富很多。习惯上，标题都使用无衬线字体、艺术字体或手写体等，而网页正文则多使用衬线字体等。

【拓展】

如果读者想要更深入地了解网页字体类型的设计方法，请扫码阅读。



3.1.2　定义字体大小



线上阅读　视频讲解



Note

使用 CSS3 的 font-size 属性可以定义字体大小，用法如下：

```
font-size: xx-small | x-small | small | medium | large | x-large | xx-large | larger | smaller | length
```

其中 xx-small（最小）、x-small（较小）、small（小）、medium（正常）、large（大）、x-large（较大）、xx-large（最大）表示绝对字体尺寸，这些特殊值将根据对象字体进行调整；larger（增大）和 smaller（减少）这对特殊值能够根据父对象中字体尺寸进行相对增大或者缩小处理，使用成比例的 em 单位进行计算；length 可以是百分数，或者是浮点数字和单位标识符组成的长度值，但不可为负值，其百分比取值基于父对象中字体的尺寸来计算，与 em 单位计算相同。

【示例】下面示例演示如何为网页定义字体大小。

启动 Dreamweaver，新建一个网页，保存为 test.html，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表。然后输入下面样式，分别设置网页字体默认大小、正文字体大小，以及栏目中字体大小。

```
body {font-size:12px;}                          /*以像素为单位设置字体大小*/
p {font-size:0.75em;}                           /*以父辈字体大小为参考设置大小*/
div {font:9pt Arial, Helvetica, sans-serif;}    /*以点为单位设置字体大小*/
```

【拓展】

如果读者想要更深入地了解网页字体大小的设计方法和各种单位的选择技巧，请扫码阅读。

3.1.3　定义字体颜色





线上阅读　视频讲解

使用 CSS3 的 color 属性可以定义字体颜色，用法如下：

```
color: color
```

参数 color 表示颜色值，取值包括颜色名、十六进制值、RGB 等颜色函数，详细说明请参考 CSS3 参考手册，或者扫码了解更详细内容。

【示例】下面示例演示如何在文档中定义字体颜色。

启动 Dreamweaver，新建一个网页，保存为 test.html，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表。然后输入下面样式，分别定义页面、段落文本、<div>标签、<span>标签包含字体颜色。

```
body {color:gray;}                       /*使用颜色名*/
p {color:#666666;}                       /*使用十六进制*/
div {color:rgb(120,120,120);}            /*使用 RGB*/
span {color:rgb(50%,50%,50%);}           /*使用 RGB*/
```

3.1.4　定义字体粗细



视频讲解

使用 CSS3 的 font-weight 属性可以定义字体粗细，用法如下：

```
font-weight: normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900
```

其中 normal 为默认值，表示正常的字体，相当于取值为 400；bold 表示粗体，相当于取值为 700，



或者使用<b >标签定义的字体效果；bolder（较粗）和 lighter（较细）是相对于 normal 字体粗细而言；另外也可以设置值为 100、200、300、400、500、600、700、800、900，它们分别表示字体的粗细，是对字体粗细的一种量化方式，值越大就表示越粗，相反就表示越细。



Note

【示例】定义字体粗细。

新建 test.html 文档，定义一个内部样式表，然后输入下面样式，分别定义段落文本、一级标题、<div>标签包含字体的粗细效果，同时定义一个粗体样式类。

```
p {font-weight: normal}                 /*等于 400*/
h1 {font-weight: 700}                   /*等于 bold*/
div{font-weight: bolder}                /*可能为 500*/
.bold {font-weight:bold;}               /*粗体样式类*/
```

注意：设置字体粗细也可以称为定义字体的重量。对于中文网页设计来说，一般仅用到 bold（加粗）、normal（普通）两个属性值。



视频讲解

3.1.5 定义艺术字体

使用 CSS3 的 font-style 属性可以定义字体倾斜效果，用法如下：

```
font-style: normal | italic | oblique
```

其中 normal 为默认值，表示正常的字体；italic 表示斜体；oblique 表示倾斜的字体。italic 和 oblique 两个取值只能在英文等西方文字中有效。

【示例】新建 test.html 文档，输入下面样式，定义一个斜体样式类。

```
.italic {/*斜体样式类*/
    font-style:italic;
}
```

在<body>标签中输入两段文本，并把斜体样式类应用到其中一段文本中。

```
<p>知我者，谓我心忧，不知我者，谓我何求。 </p>
<p class="italic">君子坦荡荡，小人长戚戚。</p>
```

最后在浏览器中预览，比较效果如图 3.2 所示。

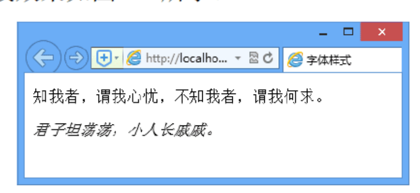


图 3.2　比较正常字体和斜体效果



视频讲解

3.1.6 定义修饰线

使用 CSS3 的 text-decoration 属性可以定义字体修饰线效果，用法如下：

```
text-decoration: none || underline || blink || overline || line-through
```

其中 normal 为默认值，表示无修饰线；underline 表示下画线效果；blink 表示闪烁效果；overline





Note

表示上画线效果；line-through 表示贯穿线效果。

【示例】设置文字修饰线。

【操作步骤】

第 1 步，新建 test.html 文档，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表。

第 2 步，输入下面样式，定义 3 个修饰字体样式类。

```
.underline {text-decoration:underline;}                /*下画线样式类*/
.overline {text-decoration:overline;}                  /*上画线样式类*/
.line-through {text-decoration:line-through;}          /*删除线样式类*/
```

第 3 步，在<body>标签中输入 3 行段落文本，并分别应用上面的修饰类样式。

```
<p class="underline">昨夜西风凋碧树，独上高楼，望尽天涯路</p>
<p class="overline">衣带渐宽终不悔，为伊消得人憔悴</p>
<p class="line-through">众里寻他千百度，蓦然回首，那人却在灯火阑珊处</p>
```

第 4 步，再定义一个样式，在该样式中，同时声明多个修饰值，定义的样式如下：

```
.line {text-decoration:line-through overline underline;}
```

第 5 步，在正文中输入一行段落文本，并把 line 样式类应该到该行文本中。

```
<p class="line">古今之成大事业、大学问者，必经过三种之境界。</p>
```

第 6 步，在浏览器中预览，多种修饰线比较效果如图 3.3 所示。

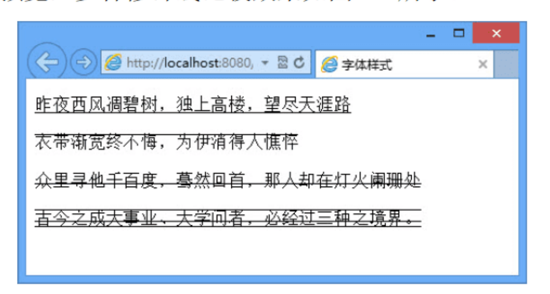


图 3.3　多种修饰线的应用效果

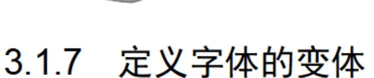
提示： CSS3 增强 text-decoration 功能，新增如下 5 个子属性。

- ☑ text-decoration-line：设置修饰线的位置，取值包括 none（无）、underline、overline、line-through、blink。
- ☑ text-decoration-color：设置修饰线的颜色。
- ☑ text-decoration-style：设置修饰线的形状，取值包括 solid、double、dotted、dashed、wavy（波浪线）。
- ☑ text-decoration-skip：设置文本修饰线条必须略过内容中的哪些部分。
- ☑ text-underline-position：设置对象中下画线的位置。

关于这些子属性的详细取值说明和用法，请参考 CSS3 参考手册。由于目前大部分浏览器暂不支持这些子属性，可以暂时忽略。


视频讲解


Note



3.1.7 定义字体的变体

使用 CSS3 的 font-variant 属性可以定义字体的变体效果，用法如下：

```
font-variant: normal | small-caps
```

其中 normal 为默认值，表示正常的字体；small-caps 表示小型的大写字母字体。

【示例】新建 test.html 文档，在内部样式表中定义一个类样式。

```
.small-caps {/*小型大写字母样式类*/
    font-variant:small-caps;}
```

然后在<body>标签中输入一行段落文本，并应用上面定义的类样式。

```
<p class="small-caps">font-variant </p>
```

注意：font-variant 仅支持拉丁字体，中文字体没有大小写效果区分。如果设置了小型大写字体，但是该字体没有找到原始小型大写字体，则浏览器会模拟一个。例如，可通过使用一个常规字体，并将其小写字母替换为缩小过的大写字母。


视频讲解

3.1.8 定义大小写字体

使用 CSS3 的 text-transform 属性可以定义字体大小写效果。用法如下：

```
text-transform: none | capitalize | uppercase | lowercase
```

其中 none 为默认值，表示无转换发生；capitalize 表示将每个单词的第一个字母转换成大写，其余无转换发生；uppercase 表示把所有字母都转换成大写；lowercase 表示把所有字母都转换成小写。

【示例】新建 test.html 文档，在内部样式表中定义 3 个类样式。

```
.capitalize {text-transform:capitalize;} /*首字母大小写样式类*/
.uppercase {text-transform:uppercase;} /*大写样式类*/
.lowercase {text-transform:lowercase;} /*小写样式类*/
```

然后在<body>标签中输入 3 行段落文本，并分别应用上面定义的类样式。

```
<p class="capitalize">text-transform:capitalize;</p>
<p class="uppercase">text-transform:uppercase;</p>
<p class="lowercase">text-transform:lowercase;</p>
```

分别在 IE 和 Firefox 浏览器中预览，则比较效果如图 3.4 和图 3.5 所示。

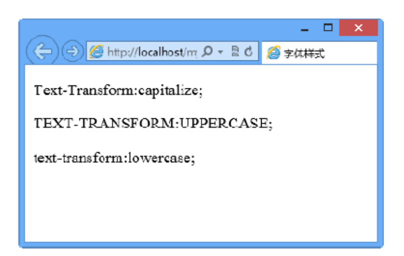


图 3.4　IE 中解析的大小效果

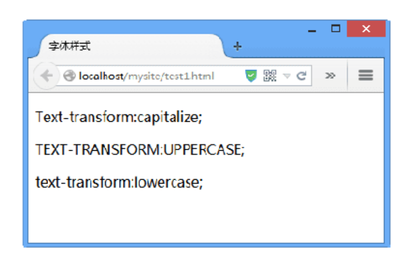


图 3.5　Firefox 中解析的大小效果

比较发现，IE 认为只要是单词就把首字母转换为大写，而 Firefox 认为只有单词通过空格间隔之



后，才能够成为独立意义上的单词，所以几个单词连在一起时就算作一个词。



3.2　设计文本样式

文本样式主要用于设计正文的排版效果，属性名以 text 为前缀进行命名，下面分别进行介绍。

3.2.1　定义文本对齐



线上阅读



视频讲解

使用 CSS3 的 text-align 属性可以定义文本的水平对齐方式，用法如下：

```
text-align: left | right | center | justify
```

其中 left 为默认值，表示左对齐；right 为右对齐；center 为居中对齐；justify 为两端对齐。

【示例】新建 test.html 文档，在内部样式表中定义 3 个对齐类样式。

```
.left {text-align: left;}
.center {text-align: center;}
.right {text-align: right;}
```

然后在<body>标签中输入 3 段文本，并分别应用这 3 个类样式。

```
<p align="left">昨夜西风凋碧树，独上高楼，望尽天涯路</p>
<p class="center">衣带渐宽终不悔，为伊消得人憔悴</p>
<p class="right">众里寻他千百度，蓦然回首，那人却在灯火阑珊处</p>
```

在浏览器中预览，比较效果如图 3.6 所示。

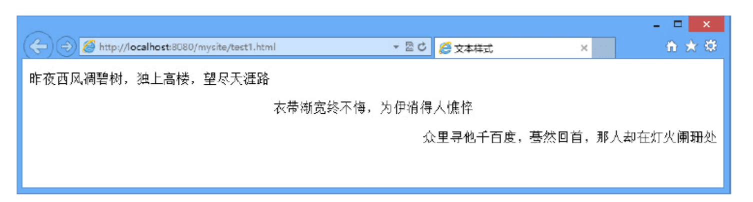


图 3.6　比较 3 种文本对齐效果

提示： CSS3 为 text-align 属性新增多个属性值，简单说明如下。

- ☑ justify：内容两端对齐（CSS2 曾经支持过，后来放弃）。
- ☑ start：内容对齐开始边界。
- ☑ end：内容对齐结束边界。
- ☑ match-parent：与 inherit（继承）表现一致。
- ☑ justify-all：效果等同于 justify，但还会让最后一行也两端对齐。

由于大部分浏览器对这些新属性值支持不是很友好，读者可以暂时忽略。

【拓展】

text-align 属性仅对行内对象有效，如文本、图像、超链接等，如果想让块元素对齐显示，如设计网页居中显示，设计<div>标签右对齐等，则需要特殊方法，感兴趣的读者可以扫码深入阅读。



3.2.2　定义垂直对齐







Note

使用 CSS3 的 vertical-align 属性可以定义文本垂直对齐，用法如下：

```
vertical-align: auto | baseline | sub | super | top | text-top | middle | bottom | text-bottom | length
```

取值简单说明如下。

- ☑ auto 将根据 layout-flow 属性的值对齐对象内容。
- ☑ baseline 为默认值，表示将支持 valign 特性的对象内容与基线对齐。
- ☑ sub 表示垂直对齐文本的下标。
- ☑ super 表示垂直对齐文本的上标。
- ☑ top 表示将支持 valign 特性的对象的内容对象顶端对齐。
- ☑ text-top 表示将支持 valign 特性的对象的文本与对象顶端对齐。
- ☑ middle 表示将支持 valign 特性的对象的内容与对象中部对齐。
- ☑ bottom 表示将支持 valign 特性的对象的内容与对象底端对齐。
- ☑ text-bottom 表示将支持 valign 特性的对象的文本与对象底端对齐。
- ☑ length 表示由浮点数字和单位标识符组成的长度值或者百分数，可为负数，定义由基线算起的偏移量，基线对于数值来说为 0，对于百分数来说就是 0%。

【示例】新建 test1.html 文档，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后输入下面样式，定义上标类样式。

```
.super {vertical-align:super;}
```

然后在<body>标签中输入一行段落文本，并应用该上标类样式。

```
<p>vertical-align 表示垂直<span class=" super ">对齐</span>属性</p>
```

在浏览器中预览，则显示效果如图 3.7 所示。

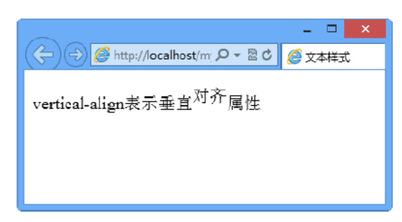


图 3.7　文本上标样式效果

【拓展】

vertical-align 属性仅对行内对象有效，如文本、图像、超链接等，如果想让块元素对齐显示，需要特殊方法，感兴趣的读者可以扫码深入阅读。



3.2.3　定义文本间距

使用 CSS3 的 letter-spacing 属性可以定义字距，使用 CSS3 的 word-spacing 属性可以定义词距。这两个属性的取值都是长度值，由浮点数字和单位标识符组成，默认值为 normal，表示默认间隔。

定义词距时，以空格为基准进行调节，如果多个单词被连在一起，则被 word-spacing 视为一个单词；如果汉字被空格分隔，则分隔的多个汉字就被视为不同的单词，word-spacing 属性此时有效。



【示例】下面示例演示如何定义字距和词距样式。

新建一个网页，保存为 test.html，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后输入下面样式，定义两个类样式。

```
.lspacing {letter-spacing:1em;}                    /*字距样式类*/
.wspacing {word-spacing:1em;}                      /*词距样式类*/
```

然后在<body>标签中输入两行段落文本，并应用上面两个类样式。

```
<p class="lspacing">letter spacing word spacing（字间距）</p>
<p class="wspacing">letter spacing word spacing（词间距）</p>
```

在浏览器中预览，则显示效果如图 3.8 所示。从图中可以直观地看到，所谓字距就是定义字母之间的间距，而词距就是定义西文单词的距离。

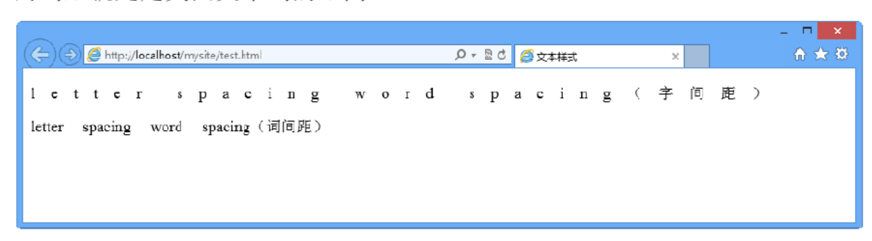


图 3.8　字距和词距演示效果比较

注意：字距和词距一般很少使用，使用时应慎重考虑用户的阅读体验和感受。对于中文用户来说，letter-spacing 属性有效，而 word-spacing 属性无效。

3.2.4　定义行高





使用 CSS3 的 line-height 属性可以定义行高，用法如下：

```
line-height : normal | length
```

其中 normal 表示默认值，一般为 1.2em；length 表示百分比数字，或者由浮点数字和单位标识符组成的长度值，允许为负值。

【示例】新建 test.html 文档，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，输入下面样式，定义两个行高类样式。

```
.p1 {/*行高样式类 1*/
    line-height:1em;                               /*行高为一个字大小*/
}
.p2 {/*行高样式类 2*/
    line-height:2em;                               /*行高为两个字大小*/
}
```

然后在<body>标签中输入两行段落文本，并应用上面两个类样式。

```
<h1>人生三境界</h1>
<h2>出自王国维《人间词话》</h2>
<p class="p1">古今之成大事业、大学问者，必经过三种之境界："昨夜西风凋碧树。独上高楼，望断天涯路。"此第一境也。"衣带渐宽终不悔，为伊消得人憔悴。"此第二境也。"众里寻他千百度，蓦然回首，那人却在灯火阑珊处。"此第三境也。此等语皆非大词人不能道。然遽以此意解释诸词，恐为晏欧诸公所不许也。</p>
```



```
<p class="p2">笔者认为，凡人都可以从容地做到第二境界，但要想逾越它却不是那么简单。成功人士果敢坚忍，不屈不挠，造就了他们不同于凡人的成功。他们逾越的不仅仅是人生的境界，更是他们自我的极限。成功后回望来路的人，才会明白另解这三重境界的话：看山是山，看水是水；看山不是山，看水不是水；看山还是山，看水还是水。</p>
```



Note

在浏览器中预览，则显示效果如图 3.9 所示。

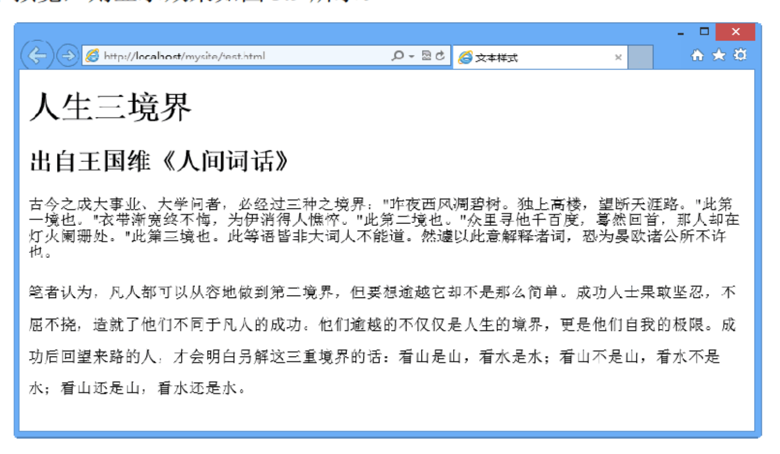


图 3.9　段落文本的行高演示效果

【拓展】

line-height 属性的用法比较复杂，灵活使用该属性，可以设计很多特殊效果，感兴趣的读者可以扫码深入阅读。



视频讲解

3.2.5　定义首行缩进

使用 CSS3 的 text-indent 属性可以定义文本首行缩进，用法如下：

```
text-indent: length
```

length 表示百分比数字或者由浮点数字和单位标识符组成的长度值，允许为负值。建议在设置缩进单位时，以 em 为设置单位，em 表示一个字距，这样可以比较精确地确定首行缩进效果。

【示例 1】使用 text-indent 属性设计首行缩进效果。

新建 test.html 文档，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，输入下面样式，定义段落文本首行缩进 2 个字距。

```
p {text-indent:2em;}                    /*首行缩进 2 个字距*/
```

然后在<body>标签中输入如下标题和段落文本。

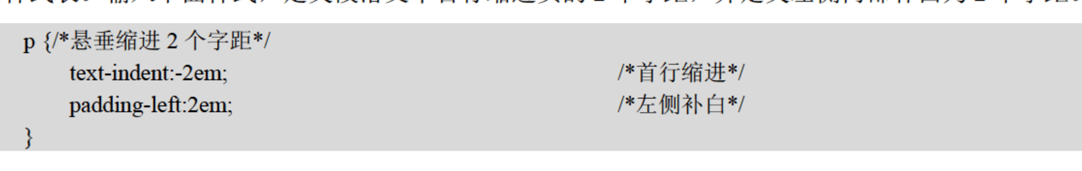
```
<h1>人生三境界</h1>
<h2>出自王国维《人间词话》</h2>
<p>古今之成大事业、大学问者，必经过三种之境界："昨夜西风凋碧树。独上高楼，望断天涯路。"此第一境也。"衣带渐宽终不悔，为伊消得人憔悴。"此第二境也。"众里寻他千百度，蓦然回首，那人却在灯火阑珊处。"此第三境也。此等语皆非大词人不能道。然遽以此意解释诸词，恐为晏欧诸公所不许也。</p>
<p>笔者认为，凡人都可以从容地做到第二境界，但要想逾越它却不是那么简单。成功人士果敢坚忍，不屈不挠，造就了他们不同于凡人的成功。他们逾越的不仅仅是人生的境界，更是他们自我的极限。成功后回望来路的人，才会明白另解这三重境界的话：看山是山，看水是水；看山不是山，看水不是水；看山还是山，看水还是水。</p>
```



在浏览器中预览，则可以看到文本缩进效果，如图 3.10 所示。

【示例 2】使用 text-indent 属性设计悬垂缩进效果。

新建一个网页，保存为 test1.html，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表。输入下面样式，定义段落文本首行缩进负的 2 个字距，并定义左侧内部补白为 2 个字距。



```
p {/*悬垂缩进 2 个字距*/
    text-indent:-2em;                 /*首行缩进*/
    padding-left:2em;                 /*左侧补白*/
}
```

Note

text-indent 属性可以取负值，定义左侧补白，防止取负值缩进导致首行文本伸到段落的边界外边。

然后在<body>标签中输入如下标题和段落文本。

```
<h1>《人间词话》节选</h1>
<h2>王国维</h2>
<p>古今之成大事业、大学问者，必经过三种之境界："昨夜西风凋碧树。独上高楼，望断天涯路。"此第一境也。"衣带渐宽终不悔，为伊消得人憔悴。"此第二境也。"众里寻他千百度，蓦然回首，那人却在灯火阑珊处。"此第三境也。此等语皆非大词人不能道。然遽以此意解释诸词，恐为晏欧诸公所不许也。</p>
<p>笔者认为，凡人都可以从容地做到第二境界，但要想逾越它却不是那么简单。成功人士果敢坚忍，不屈不挠，造就了他们不同于凡人的成功。他们逾越的不仅仅是人生的境界，更是他们自我的极限。成功后回望来路的人，才会明白另解这三重境界的话：看山是山，看水是水；看山不是山，看水不是水；看山还是山，看水还是水。</p>
```

在浏览器中预览，则可以看到文本悬垂缩进效果，如图 3.11 所示。



图 3.10　首行缩进效果



图 3.11　悬垂缩进效果

3.3　设计图像样式

在 CSS 没有普及前，主要使用<img>标签的属性来控制图像样式，如大小、边框、位置等。使用 CSS 可以更方便地控制图像显示，设计各种特殊效果，这种用法也符合 W3C 标准，是现在推荐的用法。

3.3.1　定义图像大小



视频讲解

<img>标签包含 width 和 height 属性，使用它们可以控制图像的大小。不过 CSS 提供了更符合标准的 width 和 height 属性，使用这两个属性可以更灵活地设计图像大小。





Note

【示例 1】下面是一个简单的使用 CSS 控制图像大小的示例。

启动 Dreamweaver，新建网页，保存为 test1.html，在<body>标签内输入以下代码。

```
<img class="w200" src="images/1.jpg" />
<img class="w200" src="images/2.jpg" />
<img class="w200" src="images/3.jpg" />
<img src="images/4.jpg" />
```

在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后输入下面样式，以类样式的方式控制网页中图片的显示大小。

```
.w200 {/*定义控制图像高度的类样式*/
    height:200px;
}
```

显示效果如图 3.12 所示，可以看到使用 CSS 更方便控制图片大小，提升了网页设计的灵活性。

当图像大小取值为百分比时，浏览器将根据图像包含框的宽和高进行计算。

【示例 2】本示例统一定义图像缩小 50%，然后分别放在网页中和一个固定大小的盒子中，则显示效果截然不同，比较效果如图 3.13 所示。

```
<style type="text/css">
div {/*定义固定大小的包含框*/
    height:200px;                              /*固定高度*/
    width:50%;                                 /*设计弹性宽度*/
    border:solid 1px red;                      /*定义一个边框*/
}
img {/*定义图像大小*/
    width:50%;                                 /*百分比宽度*/
    height:50%;                                /*百分比高度*/
}
</style>
<div> <img src="images/4.png" /> </div>
<img src="images/4.png" />
```

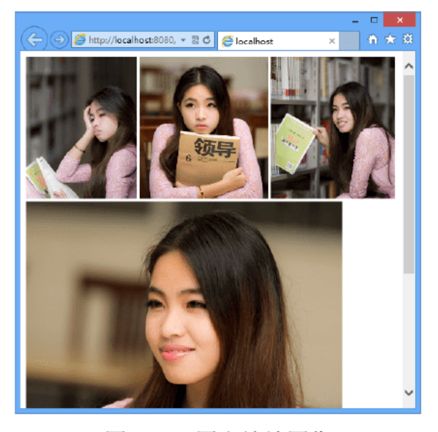


图 3.12　固定缩放图像

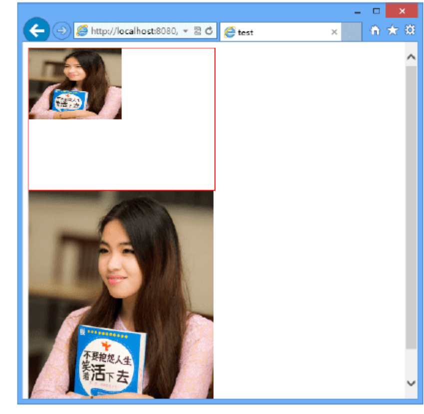


图 3.13　百分比缩放图像

提示：当仅为图像定义宽度或高度时，则浏览器能够自动调整纵横比，使宽和高能够协调缩放，避免图像变形。但是一旦同时为图像定义宽和高，就要注意宽高比，否则会失真。





3.3.2　定义图像边框

图像在默认状态下不会显示边框，但在为图像定义超链接时会自动显示 2px～3px 宽的蓝色粗边框。使用 border 属性可以清除这个边框，代码如下：

```
<a href="#"><img src="images/login.gif" alt="登录" border="0" /></a>
```

不推荐上述用法，建议使用 CSS 的 border 属性定义。CSS 的 border 属性不仅可以为图像定义边框，还提供了丰富的边框样式，支持定义边框的粗细、颜色和样式。

【示例 1】针对上面的清除图像边框效果，使用 CSS 定义则代码如下：

```
img {/*清除图像边框*/
    border:none;
}
```

使用 CSS 为<img>标签定义无边框显示，这样就不再需要为每个图像定义 0 边框的属性。下面分别讲解图像边框样式、颜色和宽度的详细用法。

1．边框样式

CSS 为元素边框定义了众多样式，边框样式可以使用 border-style 属性来定义。边框样式包括两种：虚线框和实线框。

（1）虚线框包括 dotted（点）和 dashed（虚线）。

【示例 2】在下面示例中，分别定义两个不同的点线和虚线类样式，然后分别应用到两幅图像上，效果如图 3.14 所示，通过比较可以看到点线和虚线的细微差异。

```
<style type="text/css">
img {width:250px; margin:12px;}  /*固定图像显示大小*/
.dotted {/*点线框样式类*/
    border-style:dotted;}
.dashed {/*虚线框样式类*/
    border-style:dashed;
}
</style>
<img class="dotted" src="images/1.png" alt="点线边框" />
<img class="dashed" src="images/1.png" alt="虚线边框" />
```

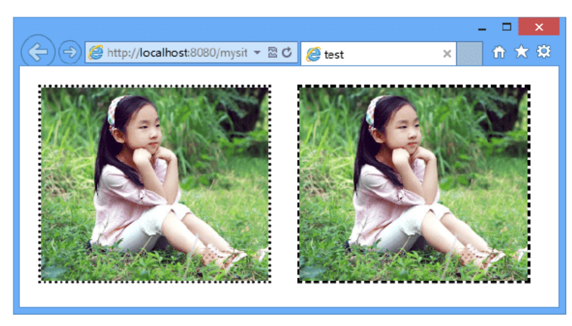


图 3.14　IE 浏览器中点线和虚线的比较效果

（2）实线框包括实线框（solid）、双线框（double）、立体凹槽（groove）、立体凸槽（ridge）、立



体凹边（inset）、立体凸边（outset）。其中实线框 solid 是应用最广的一种边框样式。



Note

提示：双线框由两条单线和中间的空隙组成，三者宽度之和等于边框的宽度。但是双线框的值分配也会存在一些矛盾，无法做到平均分配。如果边框宽度为 3px，则两条单线与其间空隙分别为 1px；如果边框宽度为 4px，则外侧单线为 2px，内侧和中间空隙分别为 1px；如果边框宽度为 5px，则两条单线宽度为 2px，中间空隙为 1px。其他取值依此类推。

2．边框颜色和宽度

使用 CSS 的 border-color 属性可以定义边框的颜色；使用 border-width 属性可以定义边框的宽度。当元素的边框样式为 none 时，所定义的边框颜色和边框宽度都会同时无效。在默认状态下，元素的边框样式为 none，而元素的边框宽度默认为 2～3px。

【示例 3】在下面示例中快速定义图像各边的边框，显示效果如图 3.15 所示。

```
<style type="text/css">
img {/*图像的边框样式*/
    width:100px;                              /*宽度*/
    border:solid red 150px;                   /*统一定义各边样式：实线框、红色、120px 宽度*/
    border-color:red blue green yellow;       /*顶边红色、右边蓝色、底边绿色、左边黄色*/
}
</style>
<img src="images/1.png" />
```

【示例 4】下面示例配合使用不同复合属性自定义各边样式，分别用 border-style、border-color 和 border-width 属性自定义图像各边边框样式，效果如图 3.16 所示。

```
<style type="text/css">
img {/*图像的边框样式*/
    width:300px;                                   /*宽度*/
    border-style:solid dashed dotted double;       /*顶边实线、右边虚线、底边点线、左边双线*/
    border-width:10px 20px 30px 40px;              /*顶边 10px、右边 20px、底边 30px、左边 40px*/
    border-color:red blue green yellow;            /*顶边红色、右边蓝色、底边绿色、左边黄色*/
}
</style>
<img src="images/1.png" />
```

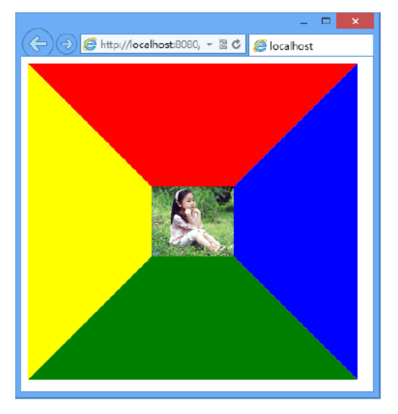


图 3.15　定义各边边框的样式效果

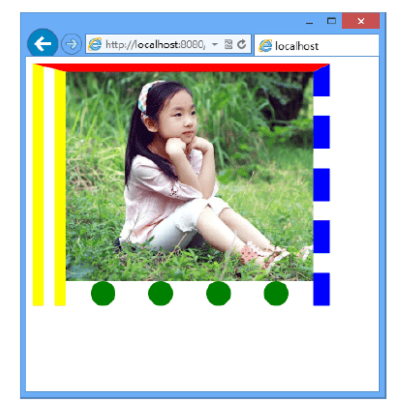


图 3.16　自定义各边边框的样式效果





如果各边样式相同，使用 border 会更方便设计。

【示例 5】下面示例定义各边样式为红色实线框，宽度为 20px。

```
div {
    width:400px;                     /*宽度*/
    height:200px;                    /*高度*/
    border:solid 20px red;           /*边框样式*/
}
```

在上面代码中，border 属性中的 3 个值分别表示边框样式、边框宽度和边框颜色，它们没有先后顺序，可以任意调整顺序。

3.3.3　定义不透明度

在 CSS3 中，使用 opacity 可以设计图像的不透明度。该属性的基本用法如下：

```
opacity:0~1;
```

参数取值范围为 0～1，数值越小，透明度越高，0 为完全透明，而 1 表示完全不透明。

提示：早期 IE 浏览器使用 CSS 滤镜定义透明度，基本用法如下：

```
filter:alpha(opacity=0~100);
```

参数取值范围为 0～100，数值越小，透明度越高，0 为完全透明，100 表示完全不透明。

【示例】在下面这个示例中，先定义一个透明度样式类，然后把它应用到图像中，并与原图进行比较，演示效果如图 3.17 所示。

```
<style type="text/css">
img {width:300px;}
.opacity {/*透明度样式类*/
    opacity: 0.3;                    /*标准用法*/
    filter:alpha(opacity=30);        /*兼容 IE 早期版本浏览器*/
    -moz-opacity:0.3;                /*兼容 Firefox 浏览器*/
}
</style>
<img src="images/1.png" title="图像不透明度" />
<img class="opacity" src="images/1.png" title="图像透明度为 0.3" />
```



图 3.17　图像透明度演示效果


视频讲解


Note

3.3.4 定义圆角特效

CSS3 新增了 border-radius 属性，使用它可以设计圆角样式。该属性用法如下：

```
border-radius:none | <length>{1,4} [/ <length>{1,4}]?;
```

border-radius 属性初始值为 none，适用于所有元素，除了 border-collapse 属性值为 collapse 的 table 元素。取值简单说明如下。

☑ none：默认值，表示元素没有圆角。

☑ <length>：由浮点数字和单位标识符组成的长度值，不可为负值。

为了方便定义元素的 4 个顶角圆角，border-radius 属性派生了 4 个子属性。

☑ border-top-right-radius：定义右上角的圆角。

☑ border-bottom-right-radius：定义右下角的圆角。

☑ border-bottom-left-radius：定义左下角的圆角。

☑ border-top-left-radius：定义左上角的圆角。

提示： border-radius 属性可包含两个参数值：第一个值表示圆角的水平半径，第二个值表示圆角的垂直半径，两个参数值通过斜线分隔。如果仅包含一个参数值，则第二个值与第一个值相同，它表示这个角是一个四分之一圆角。如果参数值包含 0，则就是矩形，不会显示为圆角。

【示例】下面示例分别设计两个圆角类样式，第一个类 r1 为固定 12px 的圆角，第二个类 r2 为弹性取值 50%的椭圆圆角，然后分别应用到不同的图像上，演示效果如图 3.18 所示。

```
<style type="text/css">
img {width:300px;border:solid 1px #eee;}
.r1 {border-radius:12px;}
.r2 {border-radius:50%;}
</style>
<img class="r1" src="images/1.png" title="圆角图像" />
<img class="r2" src="images/1.png" title="椭圆图像" />
```

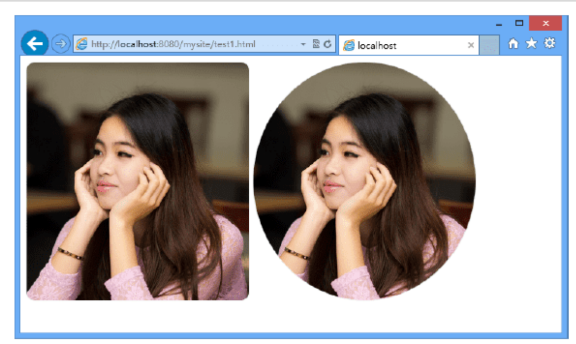


图 3.18 圆角图像演示效果


视频讲解

3.3.5 定义阴影特效

CSS3 新增了 box-shadow 属性，该属性可以定义阴影效果。该属性用法如下：

```
box-shadow:none | <shadow> [, <shadow>]*;
```



box-shadow 属性的初始值是 none，该属性适用于所有元素。取值简单说明如下。

☑ none：默认值，表示元素没有阴影。

☑ <shadow>：该属性值可以使用公式表示为 inset && [<length>{2,4} && <color>?]，其中 inset 表示设置阴影的类型为内阴影，默认为外阴影；<length>是由浮点数字和单位标识符组成的长度值，可取正值或负值，用来定义阴影水平偏移、垂直偏移，以及阴影大小（即阴影模糊度）、阴影扩展。<color>表示阴影颜色。



Note

提示：如果不设置阴影类型，默认为投影效果，当设置为 inset 时，则阴影效果为内阴影。X 轴偏移和 Y 轴偏移定义阴影的偏移距离。阴影大小、阴影扩展和阴影颜色是可选值，默认为黑色实影。box-shadow 属性值必须设置阴影的偏移值，否则没有效果。如果需要定义阴影，不需要偏移，此时可以定义阴影偏移为 0，这样才可以看到阴影效果。

【示例 1】在下面这个示例中，设计一个阴影类样式，定义圆角、阴影显示，设置圆角大小为 8px，阴影显示在右下角，模糊半径为 14px，然后分别应用到第二幅图像上，演示效果如图 3.19 所示。

```
<style type="text/css">
img {width:300px; margin:6px;}
.r1 {
    border-radius:8px;
    -moz-box-shadow:8px 8px 14px #06C;          /*兼容 Gecko 引擎*/
    -webkit-box-shadow:8px 8px 14px #06C;       /*兼容 Webkit 引擎*/
    box-shadow:8px 8px 14px #06C;               /*标准用法*/
}
</style>
<img src="images/1.png" title="无阴影图像" />
<img class="r1" src="images/1.png" title="阴影图像" />
```

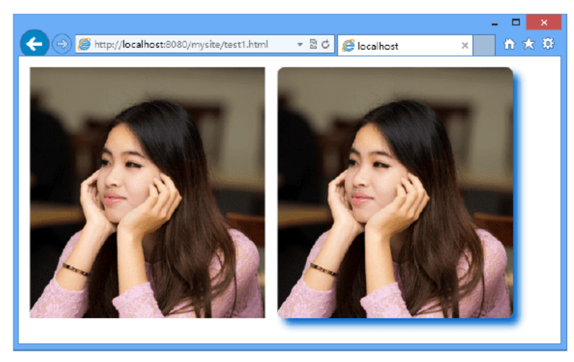


图 3.19　阴影图像演示效果

【示例 2】box-shadow 属性用法比较灵活，可以设计叠加阴影特效。例如，在上面示例中，修改类样式 r1 的代码如下，通过多组参数值定义渐变阴影效果，如图 3.20 所示。

```
img {width:300px; margin:6px;}
.r1 {
    border-radius:12px;
    box-shadow:-10px 0 12px red,
      10px 0 12px blue,
      0 -10px 12px yellow,
```



```
        0 10px 12px green;
}
```



Note

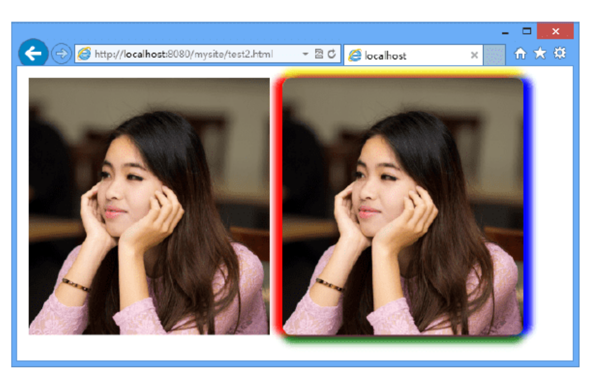


图 3.20　设计图像多层阴影效果

提示：当设计多个阴影时，需要注意书写顺序，最先写的阴影将显示在最顶层。如在上面这段代码中，先定义一个 10px 的红色阴影，再定义一个 10px 大小、10px 扩展的阴影。显示结果就是红色阴影层覆盖在黄色阴影层之上，此时如果顶层的阴影太大，就会遮盖底部的阴影。

3.4　案 例 实 战

CSS3 优化和增强了 CSS2.1 的字体和文本属性，使网页文字更具表现力和感染力，丰富了网页文本的样式和版式。



视频讲解

3.4.1　设计正文版式 1

中文版式与西文版式存在很多不同。例如，中文段落文本缩进，而西文悬垂列表；中文段落一般没有段距，而西文习惯设置一行的段距等。中文报刊文章习惯以块的适度变化来营造灵活的设计版式，中文版式中，标题习惯居中显示，正文之前喜欢设计一个题引，题引为左右缩进的段落文本显示效果，正文以首字下沉效果显示。

本案例将展示一个简单的中文版式，分别设计一级标题、二级标题、三级标题和段落文本的样式，从而使信息的轻重分明，更有利于用户阅读，演示效果如图 3.21 所示。

【操作步骤】

第 1 步，设计网页结构。本案例的 HTML 文档结构依然采用禅意花园的结构，截取第一部分的结构和内容，并把英文全部译为中文。

```
<div id="intro">
    <div id="pageHeader">
        <h1><span>CSS Zen Garden</span></h1>
        <h2><span><acronym title="cascading style sheets">CSS</acronym>设计之美</span></h2>
    </div>
    <div id="quickSummary">
        <p class="p1"><span>展示以<acronym
title="cascading style sheets">CSS</acronym>技术为基础，并提供超强的视觉冲击力。只要选择列表中任意一
```



```
个样式表，就可以将它加载到本页面中，并呈现不同的设计效果。</span></p>
            <p class="p2"><span>下载<a title="这个页面的 HTML 源代码不能够被改动。"
    href="http://www.csszengarden.com/zengarden-sample.html">HTML 文档</a> 和 <a
    title="这个页面的 CSS 样式表文件，你可以更改它。"
    href="http://www.csszengarden.com/zengarden-sample.css">CSS 文件</a>。</span></p>
        </div>
        <div id="preamble">
            <h3><span>启蒙之路</span></h3>
            <p class="p1"><span>不同浏览器随意定义标签，导致无法相互兼容的<acronym
    title="document object model">DOM</acronym>结构，或者提供缺乏标准支持的<acronym
    title="cascading style sheets">CSS</acronym>等陋习随处可见，如今当使用这些不兼容的标签和样式时，设计之路会很坎坷。</span></p>
            <p class="p2"><span>现在，我们必须清除以前为了兼容不同浏览器而使用的一些过时的小技巧。感谢<acronym
    title="world wide web consortium">W3C</acronym>、<acronym
    title="web standards project">WASP</acronym>等标准组织，以及浏览器厂家和开发师们的不懈努力，我们终于能够进入 Web 设计的标准时代。</span></p>
            <p class="p3"><span>CSS Zen Garden（样式表禅意花园）邀请你发挥自己的想象力，构思一个专业级的网页。让我们用慧眼来审视，充满理想和激情去学习 CSS 这个不朽的技术，最终使自己能够达到技术和艺术合而为一的最高境界。</span></p>
        </div>
    </div>
```



图 3.21 报刊式中文格式效果

第 2 步，定义网页基本属性。定义背景色为白色，字体为黑色。大多数浏览器默认网页就是这个样式，但是考虑到部分浏览器会以灰色背景显示，显式声明这些基本属性会更加安全。字体大小为 14px，字体为宋体。

```
    body {/*页面基本属性*/
        background:#fff;                              /*背景色*/
        color:#000;                                   /*前景色*/
        font-size:0.875em;                            /*网页字体大小*/
```



```
        font-family:"新宋体", Arial, Helvetica, sans-serif;                /*网页字体默认类型*/
    }
```

第 3 步，定义标题居中显示，适当调整标题底边距，统一为一个字距。间距设计的一般规律：字距小于行距，行距小于段距，段距小于块距。检查的方法可以尝试将网站的背景图案和线条全部去掉，看是否还能保持想要的区块感。



Note

```
    h1, h2, h3 {/*标题样式*/
        text-align:center;                                                /*居中对齐*/
        margin-bottom:1em;                                                /*定义底边界*/
    }
```

第 4 步，为二级标题定义一个下画线，并调暗字体颜色，目的是使一级标题、二级标题和三级标题在同一个中轴线显示时产生一个变化，避免单调。由于三级标题字数少（4 个汉字），可以通过适当调节字距来设计一种平衡感，避免因为字数太少而使标题看起来很单调。

```
    h2 {/*个性化二级标题样式*/
        color:#999;                                                       /*字体颜色*/
        text-decoration:underline;                                        /*下画线*/
    }
    h3 {/*个性化三级标题样式*/
        letter-spacing:0.4em;                                             /*字距*/
        font-size:1.4em;                                                  /*字体大小*/
    }
```

第 5 步，定义段落文本的样式。统一清除段落间距为 0，定义行高为 1.8 倍字体大小。

```
    p {/*统一段落文本样式*/
        margin:0;                                                         /*清除段距*/
        line-height:1.8em;                                                /*定义行高*/
    }
```

第 6 步，定义第一文本块中的第一段文本字体为深灰色，定义第一文本块中的第二段文本右对齐，定义第一文本块中的第一段和第二段文本首行缩进两个字距，同时定义第二文本块的第一段、第二段和第三段文本首行缩进两个字距。

```
    #quickSummary .p1 {/*第一文本块的第一段样式*/
        color:#444;                                                       /*字体颜色*/
    }
    #quickSummary .p2 {/*第一文本块的第二段样式*/
        text-align:right;                                                 /*右对齐*/
    }
    #quickSummary .p1, .p2, .p3 {/*除了首字下沉段以外的段样式*/
        text-indent:2em;                                                  /*首行缩进*/
    }
```

第 7 步，为第一个文本块定义左右缩进样式，设计引题的效果。

```
    #quickSummary {/*第一文本块样式*/
        margin-left:4em;                                                  /*左缩进*/
        margin-right:4em;                                                 /*右缩进*/
    }
```

第 8 步，定义首字下沉效果。CSS 提供了一个首字下沉的属性：first-letter，这是一个伪对象。什





么是伪、伪类和伪对象，我们将在超链接设计章节中进行详细讲解。但是 first-letter 属性所设计的首字下沉效果存在很多问题，所以还需要进一步设计。例如，设置段落首字浮动显示（什么是浮动请参阅 CSS 布局章节讲解），同时定义文字字号很大，以实现下沉效果。为了使首字下沉效果更明显，这里设计首字加粗、反白显示。

```
.first:first-letter {/*首字下沉样式类*/
    font-size:50px;                /*字体大小*/
    float:left;                    /*向左浮动显示*/
    margin-right:6px;              /*增加右侧边距*/
    padding:2px;                   /*增加首字四周的补白*/
    font-weight:bold;              /*加粗字体*/
    line-height:1em;               /*定义行距为一个字体大小，避免行高影响段落版式*/
    background:#000;               /*背景色*/
    color:#fff;                    /*前景色*/
}
```

注意： 由于 IE 早期版本浏览器存在 Bug，无法通过:first-letter 选择器来定义首字下沉效果，故这里重新定义了一个首字下沉的样式类（first），然后手动把这个样式类加入 HTML 文档结构对应的段落中。

```
<p class="p1 first"><span>不同浏览器随意定义标签，导致无法相互兼容的<acronym
title="document object model">DOM</acronym>结构，或者提供缺乏标准支持的<acronym
title="cascading style sheets">CSS</acronym>等陋习随处可见，如今当使用这些不兼容的标签和样式时，设计之路会很坎坷。</span></p>
```

提示： 在阅读信息时，段落文本的呈现效果多以块状存在。如果说单个字是点，一行文本为线，那么段落文本就成面了，而面以方形呈现的效率最高，网站的视觉设计大部分其实都是在拼方块。在页面版式设计中，建议坚持如下设计原则。

☑ 方块感越强，越能给用户方向感。

☑ 方块越少，越容易阅读。

☑ 方块之间以空白的形式进行分隔，从而组合为一个更大的方块。

其他样式以及整个案例效果请参阅本节实例源代码。

3.4.2　设计正文版式 2



视频讲解

本案例将展示一个简单的层级式中文版式，将一级标题、二级标题、三级标题和段落文本以阶梯状缩进，从而使信息的轻重分明，更有利于用户阅读，演示效果如图 3.22 所示。

【操作步骤】

第 1 步，复制 3.4.1 节示例源代码，删除所有的 CSS 内部样式表源代码。

第 2 步，定义页面的基本属性。这里定义页面背景色为灰绿浅色，前景色为深黑色，字体大小为 0.875em（约为 14px）。

```
body {/*页面基本属性*/
    background:#99CC99;            /*背景色*/
    color:#333333;                 /*前景色（字体颜色）*/
    margin:1em;                    /*页边距*/
    font-size:0.875em;             /*页面字体大小*/
}
```





Note



图 3.22　层级缩进式中文版式效果

第 3 步，统一标题为下画线样式，且不再加粗显示，限定上下边距为 1 个字距。在默认情况下，不同级别的标题上下边界是不同的。适当调整字距之间的疏密。

```
h1, h2, h3 {/*统一标题样式*/
    font-weight:normal;                    /*正常字体粗细*/
    text-decoration:underline;             /*下画线*/
    letter-spacing:0.2em;                  /*增加字距*/
    margin-top:1em;                        /*固定上边界*/
    margin-bottom:1em;                     /*固定下边界*/
}
```

第 4 步，分别定义不同标题级别的缩进大小，设计阶梯状缩进效果。

```
h1 {/*一级标题样式*/
    font-family:Arial, Helvetica, sans-serif;    /*标题无衬线字体*/
    margin-top:0.5em;                            /*缩小上边界*/
}
h2 {padding-left:1em;}                           /*左侧缩进 1 个字距*/
h3 {padding-left:3em;}                           /*左侧缩进 3 个字距*/
```

第 5 步，定义段落文本左缩进，同时定义首行缩进效果。清除段落默认的上下边界距离。

```
p {/*段落文本样式*/
    line-height:1.6em;                     /*行高*/
    text-indent:2em;                       /*首行缩进*/
    margin:0;                              /*清除边界*/
    padding:0;                             /*清除补白*/
    padding-left:5em;                      /*左缩进*/
}
```

3.5　在 线 练 习



在线练习

本节练习使用 CSS 设计各种文本效果、网页版式，以及文本特效。感兴趣的读者可自行扫码练习。



使用CSS3设计特效文本

CSS3优化和增强了CSS2.1的字体和文本属性，使网页文字更具表现力和感染力，丰富了网页文本的样式和版式。用户可以使用网络字体定义特殊的字体类型，摆脱了浏览器所在系统字体的局限；可以选择更多的色彩模式，创建灵活的网页配色体系；通过文本阴影，让字体看起来更美；通过动态内容，让网页内容不再单一，使CSS控制网页内容的显示能力更强。

【学习重点】

- 了解CSS3文本模块。
- 正确处理文本溢出和换行问题。
- 能够使用CSS3新色彩模式。
- 灵活定义文本阴影样式。
- 灵活添加动态内容。
- 正确使用网络字体。



4.1 CSS3 文本模块



Note

早期 CSS 文本功能就是设置字体、字号、颜色、样式、粗细、间距等。CSS3 的文本功能不再局限于这些基本设置，它优化了已经存在的各个属性，整合了各种不兼容的私有属性，给文本添加一些高级属性，并作为一个独立的模块进行开发，以尽快完善和迭代 CSS 文本功能。

4.1.1 文本模块概述

CSS3 文本模块（TextModule）把与文本相关的属性单独进行规范。文本模块的最早版本是在 2003 年制定的（http://www.w3.org/TR/2003/CR-css3-text-20030514/），2005 年对其进行了修订（http://www.w3.org/TR/2005/WD-css3-text-20050627/），2007 年又进行了系统更新（http://www.w3.org/TR/2007/WD-css3-text-20070306/），最后形成了一个较为完善的文本模型（http://www.w3.org/TR/css3-text/）。

在最终版本的文本模块中，除了新增文本属性外，还对 CSS2.1 版本中已定义的属性取值进行修补，增加了更多的属性值，以适应复杂环境中文本的呈现。与 2003 年版本相比，进行了较大的改动，其中主要改动说明如下。

- ☑ line-break 和 word-break-cjk 属性被 word-break 属性替换。
- ☑ word-break-inside 属性被 hyphenate 属性替换。
- ☑ wrap-option 属性被 text-wrap 和 word-break 属性替换。
- ☑ linefeed-treatment、white-space-treatment 和 all-space-treatment 属性被 white-space-collapse 属性替换。
- ☑ min-font-size 和 max-font-size 属性被移至下一个 CSS3 版本中字体模块内。
- ☑ 修改了 text-align 属性中 left 和 right 属性值在垂直文本中的行为。
- ☑ text-align-last 属性取消了 size 属性值。
- ☑ text-justify 属性取消了 newspaper 属性值。
- ☑ word-spacing 和 letter-spacing 属性增加了百分比取值。
- ☑ text-wrap 属性增加了 suppress 属性值。
- ☑ 删除了 linefeed-treatment 属性。
- ☑ text-align-last 属性取消了 size 属性值。
- ☑ text-justify 属性新增了 tibetan 属性值。
- ☑ punctuation-trim 属性新增加了 end 属性值。
- ☑ kerning-mode:contextual 被 punctuation-trim:adjacent 替换，其他控制被移至字体模块。
- ☑ text-shadow 属性可以继承。
- ☑ 新增 text-outline 属性。
- ☑ 新增 text-emphasis 属性，替换 font-emphasis 属性。
- ☑ 重新定义了 text-indent 属性。
- ☑ 重新设计了 hanging-punctuation 属性。

最新版本的文本模型与 2005 年版本相比，也进行了适当修订，其中增加了 text-emphasis 和 text-outline 属性，移出了 font-emphasis 属性，其他更多改动细节请参阅官方文档。





4.1.2　文本溢出

text-overflow 属性可以设置超长文本省略显示。基本语法如下：

```
text-overflow: clip | ellipsis
```

适用于块状元素，取值简单说明如下。

☑　clip：当内联内容溢出块容器时，将溢出部分裁切掉，为默认值。

☑　ellipsis：当内联内容溢出块容器时，将溢出部分替换为“...”。

提示： 在早期 W3C 文档（http://www.w3.org/TR/2003/CR-css3-text-20030514/#textoverflow-mode）中，text-overflow 被纳入规范，但是在最新修订的文档（http://www.w3.org/TR/css3-text/）中没有再包含 text-overflow 属性。

由于 W3C 规范放弃了对 text-overflow 属性的支持，所以 Mozilla 类型浏览器也放弃了对该属性的支持。不过，Mozilla developer center 推荐使用-moz-binding 的 CSS 属性进行兼容。Firefox 支持 XUL（XUL，一种 XML 的用户界面语言），这样就可以使用-moz-binding 属性来绑定 XUL 里的 ellipsis 属性了。

注意： text-overflow 属性仅是内容注解，表明当文本溢出时是否显示省略标记，并不具备样式定义的特性。要实现溢出时产生省略号的效果，还应定义两个样式：强制文本在一行内显示（white-space:nowrap）和溢出内容为隐藏（overflow:hidden），只有这样才能实现溢出文本显示省略号的效果。

【示例】下面示例设计新闻列表有序显示，对于超出指定宽度的新闻项，则使用 text-overflow 属性省略并附加省略号，避免新闻换行或者撑开版块，演示效果如图 4.1 所示。

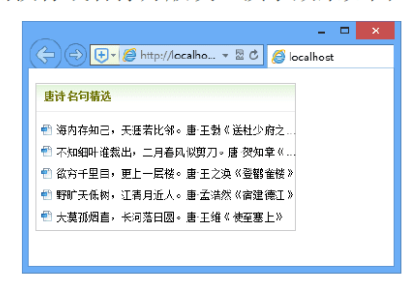


图 4.1　设计固定宽度的新闻栏目

示例代码如下：

```
<style type="text/css">
dl {/*定义新闻栏目外框，设置固定宽度*/
    width:300px;
    border:solid 1px #ccc;
}
dt {/*设计新闻栏目标题行样式*/
    padding:8px 8px;                                    /*增加文本周围空隙*/
    margin-bottom:12px;                                 /*调整底部间距*/
    background:#7FECAD url(images/green.gif) repeat-x;  /*设计标题栏背景图*/
    /*定义字体样式*/
```



```
        font-size:13px; font-weight:bold; color:#71790C;
        text-align:left;                                  /*恢复文本默认左对齐*/
        border-bottom:solid 1px #efefef;                  /*定义浅色边框线*/
    }
    dd {/*设新闻列表项样式*/
        font-size:0.78em;
        /*固定每个列表项的大小*/
        height:1.5em;width:280px;
        /*为添加新闻项目符号腾出空间*/
        padding:2px 2px 2px 18px;
        /*以背景方式添加项目符号*/
        background: url(images/icon.gif) no-repeat 6px 25%;
        margin:2px 0;
        /*为应用 text-overflow 做准备，禁止换行*/
        white-space: nowrap;
        /*为应用 text-overflow 做准备，禁止文本溢出显示*/
        overflow: hidden;
        -o-text-overflow: ellipsis;                       /*兼容 Opera*/
        text-overflow: ellipsis;                          /*兼容 IE，Safari (WebKit)*/
        -moz-binding: url('images/ellipsis.xml#ellipsis'); /*兼容 Firefox*/
    }
    </style>
    <dl>
        <dt>唐诗名句精选</dt>
        <dd>海内存知己，天涯若比邻。唐•王勃《送杜少府之任蜀州》 </dd>
        <dd>不知细叶谁裁出，二月春风似剪刀。唐•贺知章《咏柳》 </dd>
        <dd>欲穷千里目，更上一层楼。唐•王之涣《登鹳雀楼》 </dd>
        <dd>野旷天低树，江清月近人。唐•孟浩然《宿建德江》 </dd>
        <dd>大漠孤烟直，长河落日圆。唐•王维《使至塞上》 </dd>
    </dl>
```

Note


视频讲解

4.1.3 文本换行

在 CSS3 中，使用 word-break 属性可以定义文本自动换行。基本语法如下：

```
word-break: normal | keep-all | break-all
```

取值简单说明如下。

☑ normal：为默认值，依照亚洲语言和非亚洲语言的文本规则，允许在字内换行。

☑ keep-all：对于中文、韩文、日文不允许字断开。适合包含少量亚洲文本的非亚洲文本。

☑ break-all：与 normal 相同，允许非亚洲语言文本行的任意字内断开。该值适合包含一些非亚洲文本的亚洲文本，如使连续的英文字母间断行。

提示： word-break 原来是 IE 私有属性，在 CSS3 中被 Text 模块采用，得到 Chrome 和 Safari 等浏览器的支持，但不支持 keep-all 取值。

另外，IE 自定义了多个换行处理属性：line-break、word-break、word-wrap，CSS1 也定义了 white-space。这几个属性简单比较如下。

☑ line-break：指定如何（或是否）断行。除了 Firefox，其他浏览器都支持。取值说明如下所示。



- ➢ auto：使用默认的断行规则分解文本。
- ➢ loose：使用最松散的断行规则分解文本，一般用于短行的情况，如报纸。
- ➢ normal：使用最一般的断行规则分解文本。
- ➢ strict：使用最严格的断行原则分解文本。

☑ word-wrap：允许长单词或 URL 地址换行到下一行。所有浏览器都支持。取值说明如下。

- ➢ normal：只在允许的断字点换行（浏览器保持默认处理）。
- ➢ break-word：在长单词或 URL 地址内部进行换行。

☑ word-break：指定怎样在单词内断行。取值说明参考上面语法。

☑ white-space：设置如何处理元素中的空格，所有浏览器都支持。取值说明如下。

- ➢ normal：默认处理方式。
- ➢ pre：使用等宽字体显示预先格式化的文本，不合并文字间的空白距离，当文字超出边界时不换行。
- ➢ nowrap：强制在同一行内显示所有文本，合并文本间的多余空白。
- ➢ pre-wrap：使用等宽字体显示预先格式化的文本，不合并文字间的空白距离，当文字碰到边界时发生换行。
- ➢ pre-line：保持文本换行，不保留文字间的空白距离，当文字碰到边界时发生换行。

【拓展】

在 IE 浏览器下，使用“word-wrap:break-word;”声明可以确保所有文本正常显示。在 Firefox 浏览器下，中文不会出现任何问题，英文语句也不会出现问题。但是，长串英文会出现问题。为了解决长串英文问题，一般将“word-wrap:break-word;”和“word-break:break-all;”声明结合使用。但是，这种方法会导致普通英文语句中的单词被断开显示（IE 下也是）。

为了解决这个问题，可使用“word-wrap:break-word;overflow:hidden;”，而不是“word-wrap:break-word;word-break:break-all;”。“word-wrap:break-word;overflow:auto;”在 IE 下没有任何问题，但是在 Firefox 下，长串英文单词就会被遮住部分内容。

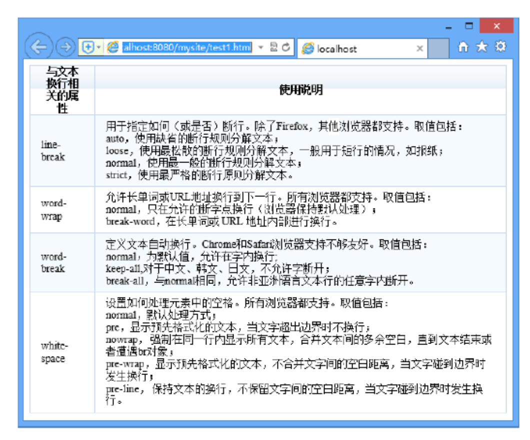
【示例】下面示例在页面中插入一个表格，由于标题行文字较多，标题行常被撑开，影响了浏览体验。为了解决这个问题，借助 CSS 换行属性进行处理，比较效果如图 4.2 所示。

```
<style type="text/css">
table {
    width: 100%;
    font-size: 14px;
    border-collapse: collapse;                  /*定义细线表格*/
    border: 1px solid #cad9ea;                  /*添加淡色细线边框*/
    table-layout: fixed;                        /*定义表格在浏览器端逐步解析呈现，避免破坏布局*/
}
th {
    background-image: url(images/th_bg1.gif);   /*使用背景图模拟渐变背景*/
    background-repeat: repeat-x;                /*定义背景图平铺方式*/
    height: 30px;
    vertical-align:middle;                      /*垂直居中显示*/
    border: 1px solid #cad9ea;                  /*添加淡色细线边框*/
    padding: 0 1em 0;
    overflow: hidden;                           /*超出范围隐藏显示，避免撑开单元格*/
    word-break: keep-all;                       /*禁止词断开显示*/
    white-space: nowrap;                        /*强迫在一行内显示*/
```





```
}
td {
    height: 20px;
    border: 1px solid #cad9ea;                /*添加淡色细线边框*/
    padding: 6px 1em;                          /*增加单元格空隙，避免文本挤在一起*/
}
tr:nth-child(even) {background-color: #f5fafe; }
.w4 {width: 4em; }
</style>
<table>
    <tr>
        <th class="w4">与文本换行相关的属性</th> <th>使用说明</th>
    </tr>
    <tr>
        <td>line-break</td><td>...</td>
    </tr>
    <tr>
        <td>word-wrap</td> <td>...</td>
    </tr>
    <tr>
        <td>word-break</td><td>...</td>
    </tr>
    <tr>
        <td>white-space</td><td>...</td>
    </tr>
</table>
```



与文本换行相关的属性	使用说明
line-break	用于指定如何（或是否）断行。除了Firefox，其他浏览器都支持。取值包括： auto，使用缺省的断行规则分解文本； loose，使用最松散的断行规则分解文本，一般用于短行的情况，如报纸； normal，使用最一般的断行规则分解文本； strict，使用最严格的断行原则分解文本。
word-wrap	允许长单词或URL地址换行到下一行。所有浏览器都支持。取值包括： normal，只在允许的断字点换行（浏览器保持默认处理）； break-word，在长单词或 URL 地址内部进行换行。
word-break	定义文本自动换行。Chrome和Safari浏览器支持不够友好。取值包括： normal，为默认值，允许在字内换行; keep-all,对于中文、韩文、日文，不允许字断开； break-all，与normal相同，允许非亚洲语言文本行的任意字内断开。
white-space	设置如何处理元素中的空格。所有浏览器都支持。取值包括： normal，默认处理方式； pre，显示预先格式化的文本，当文字超出边界时不换行； nowrap，强制在同一行内显示所有文本，合并文本间的多余空白，直到文本结束或者遭遇br对象； pre-wrap，显示预先格式化的文本，不合并文字间的空白距离，当文字碰到边界时发生换行； pre-line，保持文本的换行，不保留文字间的空白距离，当文字碰到边界时发生换行。

处理前



与文本换行	使用说明
line-break	用于指定如何（或是否）断行。除了Firefox，其他浏览器都支持。取值包括： auto，使用缺省的断行规则分解文本； loose，使用最松散的断行规则分解文本，一般用于短行的情况，如报纸； normal，使用最一般的断行规则分解文本； strict，使用最严格的断行原则分解文本。
word-wrap	允许长单词或URL地址换行到下一行。所有浏览器都支持。取值包括： normal，只在允许的断字点换行（浏览器保持默认处理）； break-word，在长单词或 URL 地址内部进行换行。
word-break	定义文本自动换行。Chrome和Safari浏览器支持不够友好。取值包括： normal，为默认值，允许在字内换行; keep-all,对于中文、韩文、日文，不允许字断开； break-all，与normal相同，允许非亚洲语言文本行的任意字内断开。
white-space	设置如何处理元素中的空格。所有浏览器都支持。取值包括： normal，默认处理方式； pre，显示预先格式化的文本，当文字超出边界时不换行； nowrap，强制在同一行内显示所有文本，合并文本间的多余空白，直到文本结束或者遭遇br对象； pre-wrap，显示预先格式化的文本，不合并文字间的空白距离，当文字碰到边界时发生换行； pre-line，保持文本的换行，不保留文字间的空白距离，当文字碰到边界时发生换行。

处理后

图 4.2　禁止表格标题文本换行显示



视频讲解

4.1.4　书写模式

CSS3 增强了文本布局中的书写模式，在 CSS2.1 定义的 direction 和 unicode-bidi 属性基础上，新增 writing-mode 属性。基本语法如下：

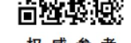
```
writing-mode: horizontal-tb | vertical-rl | vertical-lr | lr-tb | tb-rl
```



取值简单说明如下。

☑ horizontal-tb：水平方向自上而下的书写方式，类似 IE 私有值 lr-tb。
☑ vertical-rl：垂直方向自右而左的书写方式，类似 IE 私有值 tb-rl。
☑ vertical-lr：垂直方向自左而右的书写方式。
☑ lr-tb：左-右，上-下。对象中的内容在水平方向上从左向右流入，后一行在前一行的下面显示。
☑ tb-rl：上-下，右-左。对象中的内容在垂直方向上从上向下流入，自右向左。后一竖行在前一竖行的左面。全角字符是竖直向上的，半角字符（如拉丁字母或片假名）顺时针旋转 90°。

权威参考：http://www.w3.org/TR/CSS-writing-modes-3/。



权威参考

【拓展】

direction 设置文本流方向，取值包括：ltr，文本流从左到右；rtl，文本流从右到左。unicode-bidi 用于在同一个页面里定义从不同显示方向的文本，与 direction 属性一起使用。

【示例 1】下面示例设计唐诗从右侧流入，自上而下显示，效果如图 4.3 所示。

```
<style type="text/css">
#box {
    float: right;
    writing-mode: tb-rl;
    -webkit-writing-mode: vertical-rl;
    writing-mode: vertical-rl;
}
</style>
<div id="box">
    <h2>春晓</h2>
    <p>春眠不觉晓，处处闻啼鸟。夜来风雨声，花落知多少。</p>
</div>
```

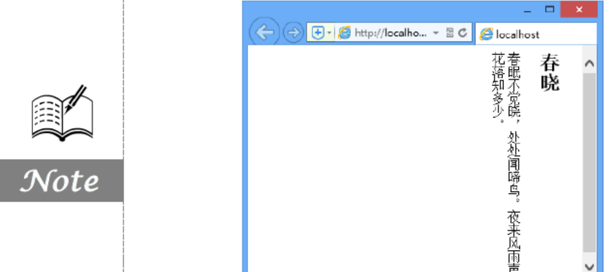
【示例 2】配合 margin-top: auto 和 margin-bottom: auto 声明，设计栏目垂直居中效果，如图 4.4 所示。

```
<style type="text/css">
.box {
    width: 400px; height: 300px;
    background-color: #f0f3f9;
    writing-mode: tb-rl;
    -webkit-writing-mode: vertical-rl;
    writing-mode: vertical-rl;
}
.auto {
    margin-top: auto;                /*垂直居中*/
    margin-bottom: auto;             /*垂直居中*/
    height:120px;
}
img {height:120px;}
</style>

<div class="box">
    <div class="auto"><img src="images/bg.png"></div>
</div>
```



Note



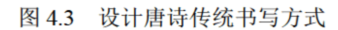


图 4.3　设计唐诗传统书写方式

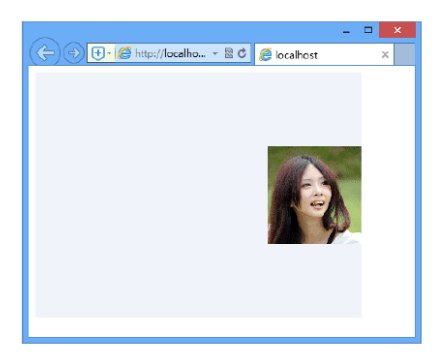


图 4.4　设计垂直居中布局

【示例 3】下面示例设计一个象棋棋子，然后定义当超链接被激活时，首行文本缩进 4px，由于使用了垂直书写模式，则文本向下移动 4px，就可以模拟一种动态下沉效果，如图 4.5 所示。

```
<style type="text/css">
.btn {
    width: 80px; height: 80px;              /*固定大小*/
    line-height: 80px;                      /*垂直居中*/
    font-size: 62px;                        /*大字体*/
    cursor: pointer;                        /*手形指针样式*/
    text-align: center;                     /*文本居中显示*/
    text-decoration:none;                   /*清除下画线*/
    color: #a78252;                         /*字体颜色*/
    background-color: #ddc390;              /*增加背景色*/
    border: 6px solid #ddc390;              /*增加粗边框*/
    border-radius: 50%;                     /*定义圆形显示*/
    /*定义阴影和内阴影边线*/
    box-shadow: inset 0 0 0 1px #d6b681, 0 1px, 0 2px, 0 3px, 0 4px;
    writing-mode: tb-rl;
    -webkit-writing-mode: vertical-rl;
    writing-mode: vertical-rl;
}
.btn:active {text-indent: 4px;}
</style>

<a href="#" class="btn">将</a>
```

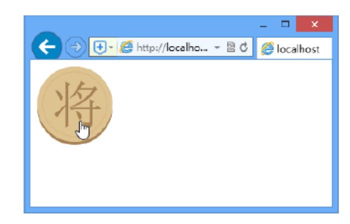


图 4.5　设计动态下沉特效



视频讲解

4.1.5　initial 值

initial 表示初始化属性的值，所有的属性都可以接受该值。如果想重置某个属性为浏览器默认设



置，那么就可以使用该值，这样就可以取消用户定义的 CSS 样式。

注意：IE 暂不支持该属性值。



【示例】在下面示例中，页面中插入了 4 段文本，然后在内部样式表中定义这 4 段文本蓝色、加粗显示，字体大小为 24px，显示效果如图 4.6 所示。

```
<style type="text/css">
p {
    color: blue;
    font-size:24px;
    font-weight:bold;
}
</style>
<p>春眠不觉晓，</p>
<p>处处闻啼鸟。</p>
<p>夜来风雨声，</p>
<p>花落知多少。</p>
```

如果想禁止第 1 句和第 3 句使用用户定义的样式，只需在内部样式表中添加一个独立样式，然后把文本样式的值都设为 initial 值即可。具体代码如下所示，运行结果如图 4.7 所示。

```
p:nth-child(odd){
    color: initial;
    font-size:initial;
    font-weight:initial;
}
```

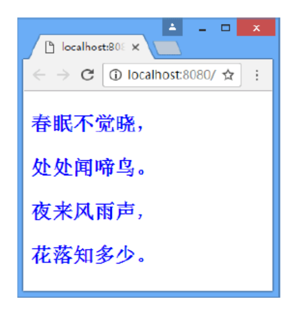


图 4.6　定义段落文本样式

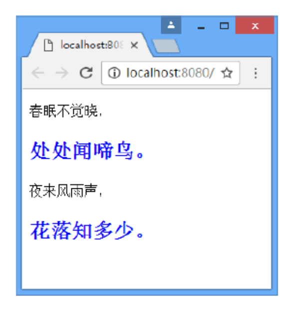


图 4.7　恢复段落文本样式

在浏览器中可以看到，第 1 句和第 3 句文本恢复为默认的黑色、常规字体，大小为 16px。

4.1.6　inherit 值



inherit 表示属性能够继承祖先的设置值，所有的属性都可以接受该值。

【示例】下面示例设置一个包含框，高度为 200px，包含两个盒子，定义盒子高度分别为 100% 和 inherit，正常情况下都会显示 200px，但是在特定情况下，例如定义盒子绝对定位显示，则设置“height:inherit;”能够按预期效果显示，而“height:100%;”就可能撑开包含框，效果如图 4.8 所示。

```
<style type="text/css">
.box {
    display: inline-block;
```



```
        height: 200px;
        width: 45%;
        border: 2px solid #666;
    }
    .box   div{
        width: 200px;
        background-color: #ccc;
        position: absolute;
    }
    .height1 {height: 100%;}
    .height2 {height: inherit;}
    </style>
    <div class="box">
        <div class="height1">height: 100%;</div>
    </div>
    <div class="box">
        <div class="height2">height: inherit;</div>
    </div>
```

Note

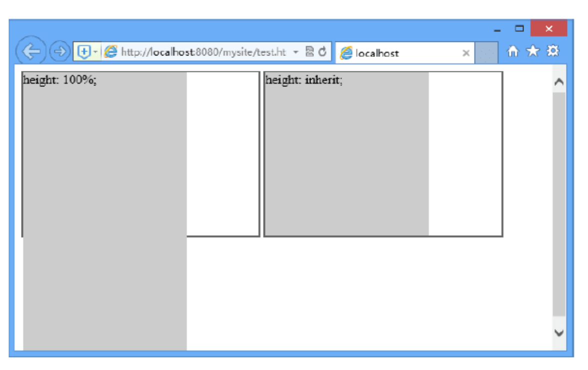


图 4.8　比较 inherit 和 100%高度效果

【补充】

inherit 表示继承属性值，一般用于字体、颜色、背景等；auto 表示自适应，一般用于高度、宽度、外边距和内边距等关于长度的属性。



视频讲解

4.1.7　unset 值

unset 表示擦除用户声明的属性值，所有的属性都可以接受该值。如果属性有继承的值，则该属性的值等同于 inherit，即继承的值不被擦除；如果属性没有继承的值，则该属性的值等同于 initial，即擦除用户声明的值，恢复初始值。

注意：IE 和 Safari 暂时不支持该属性值。

【示例】下面示例设计 4 段文本，第 1 段和第 2 段位于<div class="box">容器中，设置段落文本显示为 30px 的蓝色字体，现在擦除第 2 段和第 4 段文本样式，则第 2 段文本显示继承样式，即 12px 的红色字体，而第 4 段文本显示初始化样式，即 16px 的黑色字体，效果如图 4.9 所示。

```
<style type="text/css">
.box {color: red; font-size: 12px;}
```






```
p {color: blue; font-size: 30px;}
p.unset {
    color: unset;
    font-size: unset;
}
</style>
<div class="box">
    <p>春眠不觉晓，</p>
    <p class="unset">处处闻啼鸟。</p>
</div>
<p>夜来风雨声，</p>
<p class="unset">花落知多少。</p>
```

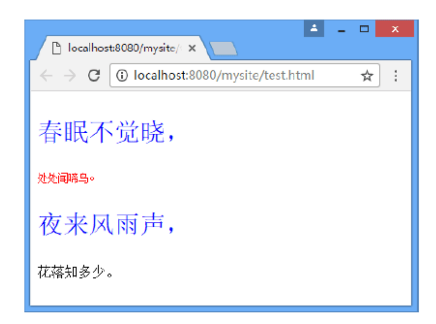


图 4.9　比较擦除后文本效果

4.1.8　all 属性

视频讲解

all 属性表示 CSS 的所有属性，但不包括 unicode-bidi 和 direction 这两个 CSS 属性。

注意：IE 暂时不支持该属性。

【示例】针对 4.1.7 节示例，我们可以简化 p.unset 类样式。

```
p.unset {
    all: unset;
}
```

如果在样式中声明的属性非常多，使用 all 会极为方便，可以避免逐个设置每个属性。

4.2　色彩模式

CSS2.1 支持 Color Name（颜色名称）、HEX（十六进制颜色值）、RGB，CSS3 新增 3 种颜色模式：RGBA、HSL 和 HSLA，下面分别进行介绍。

权威参考：http://www.w3.org/TR/css3-color/。



4.2.1　rgba()函数

RGBA 是 RGB 色彩模式的扩展，它在红、绿、蓝三原色通道基础上增加了 Alpha 通道。其语法



格式如下：

```
rgba(r,g,b,<opacity>)
```



Note

参数说明如下。

☑ r、g、b：分别表示红色、绿色、蓝色三原色所占的比重。取值为正整数或者百分数。正整数值的取值范围为0～255，百分数值的取值范围为0.0%～100.0%。超出范围的数值将被截至其最接近的取值极限。注意，并非所有浏览器都支持使用百分数值。

☑ <opacity>：表示不透明度，取值范围为0～1。

【示例】 下面示例使用CSS3的box-shadow属性和rgba()函数为表单控件设置半透明度的阴影，来模拟柔和的润边效果。示例主要代码如下，预览效果如图4.10所示。

```
<style type="text/css">
input, textarea {/*统一文本框样式*/
    padding: 4px;                        /*增加内补白，增大表单对象尺寸，看起来更大方*/
    border: solid 1px #E5E5E5;           /*增加淡淡的边框线*/
    outline: 0;                          /*清除轮廓线*/
    font: normal 13px/100% Verdana, Tahoma, sans-serif;
    width: 200px;                        /*固定宽度*/
    background: #FFFFFF;                 /*白色背景*/
    /*设置边框阴影效果*/
    box-shadow: rgba(0, 0, 0, 0.1) 0px 0px 8px;
}
/*定义表单对象获取焦点、鼠标经过时，高亮显示边框*/
input:hover, textarea:hover, input:focus, textarea:focus {border-color: #C9C9C9;}
label {/*定义标签样式*/
    margin-left: 10px;
    color: #999999;
    display:block; /*以块状显示，实现分行显示*/
}
.submit input {/*定义提交按钮样式*/
    width:auto;                          /*自动调整宽度*/
    padding: 9px 15px;                   /*增大按钮尺寸，看起来更大气*/
    background: #617798;                 /*设计扁平化单色背景*/
    border: 0;                           /*清除边框线*/
    font-size: 14px;                     /*固定字体大小*/
    color: #FFFFFF;                      /*白色字体*/
}
</style>
<form>
    <p class="name">
        <label for="name">姓名</label>
        <input type="text" name="name" id="name" />
    </p>
    <p class="email">
        <label for="email">邮箱</label>
        <input type="text" name="email" id="email" />
    </p>
    <p class="submit">
        <input type="submit" value="提交" />
```



```
    </p>
</form>
```

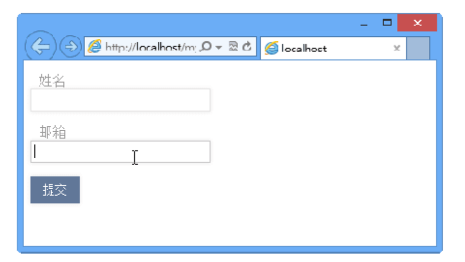
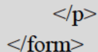


图 4.10　设计带有阴影边框的表单效果




提示：rgba(0,0,0,0.1)表示不透明度为 0.1 的黑色，这里不宜直接设置为浅灰色，因为对于非白色背景来说，灰色发虚，而半透明效果可以避免这种情况。

4.2.2　hsl()函数


视 频 讲 解

HSL 是一种标准的色彩模式，包括了人类视力所能感知的所有颜色，在屏幕上可以重现 16777216 种颜色，是目前运用最广泛的颜色系统。它通过色调（H）、饱和度（S）和亮度（L）3 个颜色通道的叠加来获取各种颜色。其语法格式如下：

```
hsl(<length>,<percentage>,<percentage>)
```

参数说明如下。

☑　<length>：表示色调（Hue）。可以为任意数值，用以确定不同的颜色。其中 0（或 360、-360）表示红色，60 表示黄色，120 表示绿色，180 表示青色，240 表示蓝色，300 表示洋红。

☑　<percentage>（第一个）：表示饱和度（Saturation），取值范围为 0～100%。其中 0%表示灰度，即没有使用该颜色；100%饱和度最高，即颜色最艳。

☑　<percentage>（第二个）：表示亮度（Lightness），取值范围为 0～100%。其中 0%最暗，显示为黑色；50%表示均值；100%最亮，显示为白色。

【示例】设计颜色表。先选择一个色值，然后通过调整颜色的饱和度和亮度比重，分别设计不同的配色方案表。在网页设计中，利用这种方法可以根据网页需要选择恰当的配色方案。使用 HSL 颜色表现方式，可以很轻松地设计网页配色方案表，模拟演示效果如图 4.11 所示。

```
<styletype="text/css">
/*设计表格边框样式并增加内部间距，以方便观看*/
table{border: solid 1px red; background:#eee; padding:6px;}
/*设计列标题字体样式*/
th{color:red; font-size:12px; font-weight:normal;}
/*设计单元格大小尺寸*/
td{width:80px; height:30px;}
/*第 1 行*/
tr:nth-child(4) td:nth-of-type(1){background:hsl(0,100%,100%);}/*第 1 列*/
tr:nth-child(4) td:nth-of-type(2){background:hsl(0,75%,100%);}/*第 2 列*/
tr:nth-child(4) td:nth-of-type(3){background:hsl(0,50%,100%);}/*第 3 列*/
tr:nth-child(4) td:nth-of-type(4){background:hsl(0,25%,100%);}/*第 4 列*/
```



Note

```
tr:nth-child(4) td:nth-of-type(5){background:hsl(0,0%,100%);}/*第 5 列*/
/*第 2 行*/
tr:nth-child(5) td:nth-of-type(1){background:hsl(0,100%,88%);}/*第 1 列*/
tr:nth-child(5) td:nth-of-type(2){background:hsl(0,75%,88%);}/*第 2 列*/
tr:nth-child(5) td:nth-of-type(3){background:hsl(0,50%,88%);}/*第 3 列*/
tr:nth-child(5) td:nth-of-type(4){background:hsl(0,25%,88%);}/*第 4 列*/
tr:nth-child(5) td:nth-of-type(5){background:hsl(0,0%,88%);}/*第 5 列*/
/*第 3 行*/
tr:nth-child(6) td:nth-of-type(1){background:hsl(0,100%,75%);}/*第 1 列*/
tr:nth-child(6) td:nth-of-type(2){background:hsl(0,75%,75%);}/*第 2 列*/
tr:nth-child(6) td:nth-of-type(3){background:hsl(0,50%,75%);}/*第 3 列*/
tr:nth-child(6) td:nth-of-type(4){background:hsl(0,25%,75%);}/*第 4 列*/
tr:nth-child(6) td:nth-of-type(5){background:hsl(0,0%,75%);}/*第 5 列*/
/*第 4 行*/
tr:nth-child(7) td:nth-of-type(1){background:hsl(0,100%,63%);}/*第 1 列*/
tr:nth-child(7) td:nth-of-type(2){background:hsl(0,75%,63%);}/*第 2 列*/
tr:nth-child(7) td:nth-of-type(3){background:hsl(0,50%,63%);}/*第 3 列*/
tr:nth-child(7) td:nth-of-type(4){background:hsl(0,25%,63%);}/*第 4 列*/
tr:nth-child(7) td:nth-of-type(5){background:hsl(0,0%,63%);}/*第 5 列*/
/*第 5 行*/
tr:nth-child(8) td:nth-of-type(1){background:hsl(0,100%,50%);}/*第 1 列*/
tr:nth-child(8) td:nth-of-type(2){background:hsl(0,75%,50%);}/*第 2 列*/
tr:nth-child(8) td:nth-of-type(3){background:hsl(0,50%,50%);}/*第 3 列*/
tr:nth-child(8) td:nth-of-type(4){background:hsl(0,25%,50%);}/*第 4 列*/
tr:nth-child(8) td:nth-of-type(5){background:hsl(0,0%,50%);}/*第 5 列*/
/*第 6 行*/
tr:nth-child(9) td:nth-of-type(1){background:hsl(0,100%,38%);}/*第 1 列*/
tr:nth-child(9) td:nth-of-type(2){background:hsl(0,75%,38%);}/*第 2 列*/
tr:nth-child(9) td:nth-of-type(3){background:hsl(0,50%,38%);}/*第 3 列*/
tr:nth-child(9) td:nth-of-type(4){background:hsl(0,25%,38%);}/*第 4 列*/
tr:nth-child(9) td:nth-of-type(5){background:hsl(0,0%,38%);}/*第 5 列*/
/*第 7 行*/
tr:nth-child(10) td:nth-of-type(1){background:hsl(0,100%,25%);}/*第 1 列*/
tr:nth-child(10) td:nth-of-type(2){background:hsl(0,75%,25%);}/*第 2 列*/
tr:nth-child(10) td:nth-of-type(3){background:hsl(0,50%,25%);}/*第 3 列*/
tr:nth-child(10) td:nth-of-type(4){background:hsl(0,25%,25%);}/*第 4 列*/
tr:nth-child(10) td:nth-of-type(5){background:hsl(0,0%,25%);}/*第 5 列*/
/*第 8 行*/
tr:nth-child(11) td:nth-of-type(1){background:hsl(0,100%,13%);}/*第 1 列*/
tr:nth-child(11) td:nth-of-type(2){background:hsl(0,75%,13%);}/*第 2 列*/
tr:nth-child(11) td:nth-of-type(3){background:hsl(0,50%,13%);}/*第 3 列*/
tr:nth-child(11) td:nth-of-type(4){background:hsl(0,25%,13%);}/*第 4 列*/
tr:nth-child(11) td:nth-of-type(5){background:hsl(0,0%,13%);}/*第 5 列*/
/*第 9 行*/
tr:nth-child(12) td:nth-of-type(1){background:hsl(0,100%,0%);}/*第 1 列*/
tr:nth-child(12) td:nth-of-type(2){background:hsl(0,75%,0%);}/*第 2 列*/
tr:nth-child(12) td:nth-of-type(3){background:hsl(0,50%,0%);}/*第 3 列*/
tr:nth-child(12) td:nth-of-type(4){background:hsl(0,25%,0%);}/*第 4 列*/
tr:nth-child(12) td:nth-of-type(5){background:hsl(0,0%,0%);}/*第 5 列*/
</style>
```



```
<table class="hslexample">
    <tbody>
        <tr>
            <th> </th><th colspan="5">色相：H=0 Red </th>
        </tr>
        <tr>
            <th> </th><th colspan="5">饱和度 (&rarr;)</th>
        </tr>
        <tr>
            <th>亮度 (&darr;)</th>
            <th>100% </th><th>75% </th><th>50% </th><th>25% </th><th>0% </th>
        </tr>
        …
    </tbody>
</table>
```





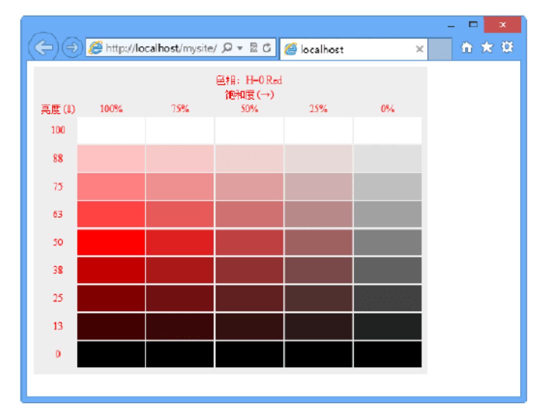


图 4.11　使用 HSL 颜色值设计颜色表

在上面代码中，tr:nth-child(4) td:nth-of-type(1)中的 tr:nth-child(4)子选择器表示选择行，而 td:nth-of-type(1)表示选择单元格（列）。其他行选择器结构依此类推。在“background:hsl(0,0%,0%);”声明中，hsl()函数的第 1 个参数值 0 表示色相值，第 2 个参数值 0%表示饱和度，第 3 个参数值 0%表示亮度。

4.2.3　hsla()函数



视 频 讲 解

HSLA 是 HSL 色彩模式的扩展，在色相、饱和度、亮度三要素基础上增加了不透明度参数。使用 HSLA 色彩模式，可以定义不同透明效果。其语法格式如下：

```
hsla(<length>,<percentage>,<percentage>,<opacity>)
```

其中前 3 个参数与 hsl()函数参数含义和用法相同，第 4 个参数<opacity>表示不透明度，取值范围为 0～1。

【示例】下面示例设计一个简单的登录表单，表单对象的边框色使用#fff 值进行设置，定义为白色；表单对象的阴影色使用 rgba(0,0,0,0.1)值进行设置，定义为非常透明的黑色；字体颜色使用 hsla(0,0%,100%,0.9)值进行设置，定义为轻微透明的白色。预览效果如图 4.12 所示。

```
<style  type="text/css">
body{/*为页面添加背景图像，显示在中央顶部位置，并列完全覆盖窗口*/
```





Note

```
        background: #eedfcc url(images/bg.jpg) no-repeat center top;
        background-size: cover;
}
.form {/*定义表单框的样式*/
        width: 300px;                              /*固定表单框的宽度*/
        margin: 30px auto;                         /*居中显示*/
        border-radius: 5px;                        /*设计圆角效果*/
        box-shadow: 0 0 5px rgba(0,0,0,0.1),       /*设计润边效果*/
                    0 3px 2px rgba(0,0,0,0.1);     /*设计淡淡的阴影效果*/
}
.form p {/*定义表单对象外框圆角、白边显示*/
        width: 100%;
        float: left;
        border-radius: 5px;
        border: 1px solid #fff;
}
/*定义表单对象样式*/
.form input[type=text],
.form input[type=password] {
        /*固定宽度和大小*/
        width: 100%;
        height: 50px;
        padding: 0;
        /*增加修饰样式*/
        border: none;                              /*移除默认的边框样式*/
        background: rgba(255,255,255,0.2);         /*增加半透明的白色背景*/
        box-shadow: inset 0 0 10px rgba(255,255,255,0.5);  /*为表单对象设计高亮效果*/
        /*定义字体样式*/
        text-indent: 10px;
        font-size: 16px;
        color:hsla(0,0%,100%,0.9);
        text-shadow: 0 -1px 1px rgba(0,0,0,0.4);   /*为文本添加阴影，设计立体效果*/
}
.form input[type=text] {                           /*设计用户名文本框底部边框样式，并设计顶部圆角*/
        border-bottom: 1px solid rgba(255,255,255,0.7);
        border-radius: 5px 5px 0 0;
}
.form input[type=password] {                       /*设计密码域文本框顶部边框样式，并设计底部圆角*/
        border-top: 1px solid rgba(0,0,0,0.1);
        border-radius: 0 0 5px 5px;
}
/*定义表单对象被激活或者鼠标经过时，增亮背景色，并清除轮廓线*/
.form input[type=text]:hover,
.form input[type=password]:hover,
.form input[type=text]:focus,
.form input[type=password]:focus {
        background: rgba(255,255,255,0.4);
        outline: none;
}
</style>
```



```
<form class="form">
    <p>
        <input type="text" id="login" name="login" placeholder="用户名">
        <input type="password" name="password" id="password" placeholder="密码">
    </p>
</form>
```



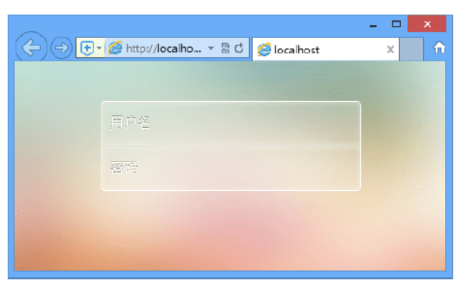


图 4.12　设计登录表单

4.2.4　opacity 属性



opacity 属性定义元素对象的不透明度。其语法格式如下：

```
opacity: <alphavalue> | inherit;
```

取值简单说明如下。

☑ <alphavalue>：由浮点数字和单位标识符组成的长度值。不可为负值，默认值为 1。opacity 取值为 1 时，元素是完全不透明的；取值为 0 时，元素是完全透明、不可见的；介于 1 到 0 之间的任何值都表示该元素的不透明程度。如果超过了这个范围，其计算结果将截取到与之最相近的值。

☑ inherit：表示继承父辈元素的不透明性。

【示例】下面示例设计<div class="bg">对象铺满整个窗口，显示为黑色背景，不透明度为 0.7，这样可以模拟一种半透明的遮罩效果；再使用 CSS 定位属性设计<div class="login">对象显示在上面。示例主要代码如下，演示效果如图 4.13 所示。

```
<style type="text/css">
body {margin: 0; padding: 0;}
div {position: absolute;}
.bg {
    width: 100%;
    height: 100%;
    background: #000;
    opacity: 0.7;
    filter: alpha(opacity=70);}
. login {
    text-align:center;
    width:100%;
    top: 20%;
}
</style>
```



```
<div class="web"><img src="images/bg.png" /></div>
<div class="bg"></div>
<div class="login"><img src="images/login.png"   /></div>
```



Note

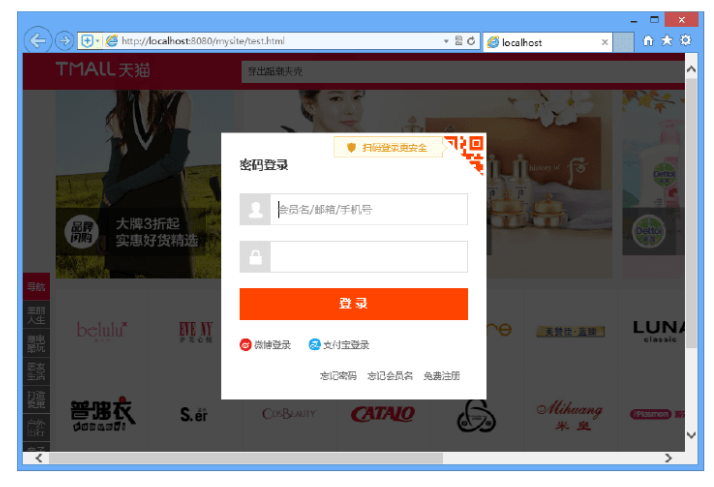


图 4.13　设计半透明的背景布效果

注意：使用色彩模式函数的 Alpha 通道可以针对元素的背景色或文字颜色单独定义不透明度，而 opacity 属性只能为整个对象定义不透明度。



视 频 讲 解

4.2.5　transparent 值

transparent 属性值用来指定全透明色彩，等效于 rgba(0,0,0,0)值。

【示例】下面示例使用 CSS 的 border 设计三角形效果，通过 transparent 颜色值让部分边框透明显示。代码如下所示，效果如图 4.14 所示。

```
<style type="text/css">
#demo {
    width: 0; height: 0;
    border-left: 50px solid transparent;
    border-right: 50px solid transparent;
    border-bottom: 100px solid red;
}
</style>
<div id="demo"></div>
```

通过调整各边颜色设置或者各边宽度，可以设计不同角度的三角形或直角等不同形状。

（1）设计向右三角形

```
#demo {
    width: 0; height: 0;
    border-top: 50px solid transparent;
    border-left: 100px solid red;
    border-bottom: 50px solid transparent;
}
```



（2）设计直角三角形

```
#demo {
    width: 0; height: 0;
    border-top: 100px solid red;
    border-right: 100px solid transparent;
}
```

（3）设计梯形（效果如图 4.15 所示）

```
#demo {
    height: 0;
    width: 120px;
    border-bottom: 120px solid #ec3504;
    border-left: 60px solid transparent;
    border-right: 60px solid transparent;
}
```



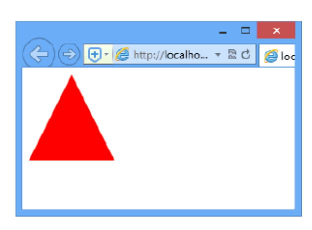


图 4.14　设计三角形效果

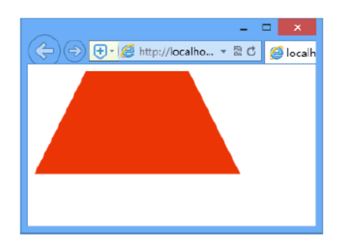


图 4.15　设计梯形效果

4.2.6　currentColor 值



视频讲解

在 CSS 中，border-color、box-shadow 和 text-decoration-color 属性的默认值是 color 属性的值。

【示例 1】在下面示例中，为段落文本增加边框线，边框线的颜色为“color:red;”，显示为红色。

```
<style type="text/css">
p {
    border:solid 2px;
    color:red;
}
</style>
<p>春眠不觉晓，处处闻啼鸟。夜来风雨声，花落知多少。</p>
```

在 CSS1 和 CSS2 中，却没有为此定义一个相应的关键字。为此 CSS3 扩展了颜色值，包含 currentColor 关键字，并用于所有接受颜色的属性上。currentColor 表示 color 属性的值。

【示例 2】在下面示例中，设计图标背景颜色值为 currentColor，这样在网页中随着链接文本的字体颜色不断变化，图标的颜色也跟随链接文本的颜色变化而变化，确保整体导航条色彩一致性，达到图文合一的境界，效果如图 4.16 所示。

```
<style type="text/css">
.icon {
    display: inline-block;
    width: 16px; height: 20px;
    background-image: url(images/sprite_icons.png);
```



Note

```
        background-color: currentColor; /*使用当前颜色控制图标的颜色*/
    }
    .icon1 {background-position: 0 0;}
    .icon2 {background-position: -20px 0;}
    .icon3 {background-position: -40px 0;}
    .icon4 {background-position: -60px 0;}
    .link {margin-right: 15px;}
    .link:hover {color: red;}/*虽然改变的是文字颜色，但是图标颜色也一起变化了*/
</style>
<a href="##" class="link"><i class="icon icon1"></i>首页</a>
<a href="##" class="link"><i class="icon icon2"></i>刷新</a>
<a href="##" class="link"><i class="icon icon3"></i>收藏</a>
<a href="##" class="link"><i class="icon icon4"></i>展开</a>
```

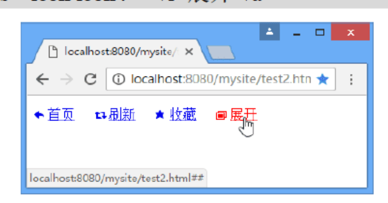


图 4.16　设计图标背景色为 currentColor

提示：如果将 color 属性设置为 currentColor，则相当于 color: inherit。

4.3　文本阴影

使用 text-shadow 属性可以为文本添加阴影效果，截至目前，Safari、Firefox、Chrome 和 Opera 等主流浏览器都支持该功能。

4.3.1　定义 text-shadow



视频讲解

text-shadow 属性是在 CSS2 中定义的，在 CSS2.1 中被删除，在 CSS3 的 Text 模块中又恢复。基本语法如下：

```
text-shadow: none | <length>{2,3} && <color>?
```

取值简单说明如下。

☑ none：无阴影，为默认值。

☑ <length>①：第 1 个长度值用来设置对象的阴影水平偏移值。可以为负值。

☑ <length>②：第 2 个长度值用来设置对象的阴影垂直偏移值。可以为负值。

☑ <length>③：如果提供了第 3 个长度值则用来设置对象的阴影模糊值。不允许为负值。

☑ <color>：设置对象阴影的颜色。

【示例】下面为段落文本定义一个简单的阴影效果，演示效果如图 4.17 所示。

```
<style type="text/css">
p {
    text-align: center;
```



```
        font: bold 60px helvetica, arial, sans-serif;
        color: #999;
        text-shadow: 0.1em 0.1em #333;
    }
    </style>
    <p>HTML5+CSS3</p>
```





图 4.17　定义文本阴影

“text-shadow: 0.1em 0.1em #333;”声明了右下角文本阴影效果，如果把投影设置到左上角，则可以这样声明：

```
p {text-shadow: -0.1em -0.1em #333;}
```

效果如图 4.18 所示。

同理，如果设置阴影在文本的左下角，则可以设置如下样式：

```
p {text-shadow: -0.1em 0.1em #333;}
```

演示效果如图 4.19 所示。

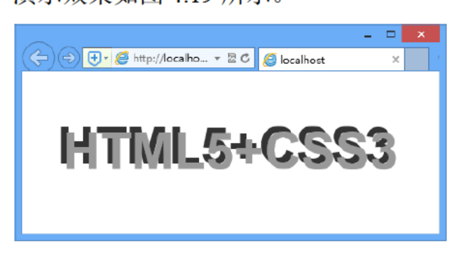


图 4.18　定义左上角阴影



图 4.19　定义左下角阴影

也可以增加模糊效果的阴影，效果如图 4.20 所示。

```
p{text-shadow: 0.1em 0.1em 0.3em #333;}
```

或者定义如下模糊阴影效果，效果如图 4.21 所示。

```
p{text-shadow: 0.1em 0.1em 0.2em black;}
```



图 4.20　增加模糊阴影



图 4.21　定义模糊阴影





提示：在 text-shadow 属性的第一个值和第二个值中，正值偏右或偏下，负值偏左或偏上。在阴影偏移之后，可以指定一个模糊半径。模糊半径是个长度值，指出模糊效果的范围。如何计算模糊效果的具体算法并没有指定。在阴影效果的长度值之前或之后还可以选择指定一个颜色值。颜色值会被用作阴影效果的基础。如果没有指定颜色，那么将使用 color 属性值来替代。



视频讲解

4.3.2 案例：设计特效字

下面结合示例介绍如何灵活使用 text-shadow 属性设计特效文字效果。

【示例 1】下面示例通过阴影把文本颜色与背景色区分开来，让字体看起来更清晰。代码如下，演示效果如图 4.22 所示。

```
<style type="text/css">
p {
    text-align: center;
    font: bold 60px helvetica, arial, sans-serif;
    color: #fff;
    text-shadow: black 0.1em 0.1em 0.2em;
}
</style>
<p>HTML5+CSS3</p>
```

【示例 2】下面示例演示了如何为红色文本定义 3 个不同颜色的阴影，演示效果如图 4.23 所示。



图 4.22　使用阴影增加前景色和背景色对比度

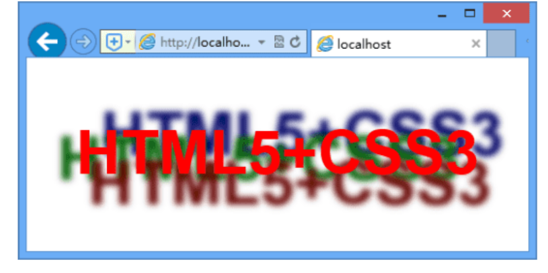


图 4.23　定义多色阴影

当使用 text-shadow 属性定义多色阴影时，每个阴影效果必须指定阴影偏移，而模糊半径、阴影颜色是可选参数。

```
<style type="text/css">
p {
    text-align: center;
    font:bold 60px helvetica, arial, sans-serif;
    color: red;
    text-shadow: 0.2em 0.5em 0.1em #600,
        -0.3em 0.1em 0.1em #060,
        0.4em -0.3em 0.1em #006;
}
</style>
<p>HTML5+CSS3</p>
```





> 提示：text-shadow 属性可以接受以逗号分隔的阴影效果列表，并应用到该元素的文本上。阴影效果按照给定的顺序应用，因此可能出现互相覆盖，但是它们永远不会覆盖文本本身。阴影效果不会改变框的尺寸，但可能延伸到它的边界之外。阴影效果的堆叠层次和元素本身的层次是一样的。

【示例 3】下面示例演示把阴影设置到文本线框的外面。代码如下，演示效果如图 4.24 所示。

```
<style type="text/css">
p {
    text-align: center;
    font:bold 60px helvetica, arial, sans-serif;
    color: red;
    border:solid 1px red;
    text-shadow: 0.5em 0.5em 0.1em #600,
        -1em 1em 0.1em #060,
        0.8em -0.8em 0.1em #006;
}
</style>
<p>HTML5+CSS3</p>
```

【示例 4】下面示例借助阴影效果列表机制，使用阴影叠加出燃烧的文字特效。代码如下，演示效果如图 4.25 所示。

```
<style type="text/css">
body {background:#000;}
p {
    text-align: center;
    font:bold 60px helvetica, arial, sans-serif;
    color: red;
    text-shadow: 0 0 4px white,
        0 -5px 4px #ff3,
        2px -10px 6px #fd3,
        -2px -15px 11px #f80,
        2px -25px 18px #f20;
}
</style>
<p>HTML5+CSS3</p>
```



图 4.24　将阴影设置到文本线框外面



图 4.25　定义燃烧的文字特效

【示例 5】text-shadow 属性可以使用在:first-letter 和:first-line 伪元素上。同时，还可以利用该属性设计立体文本。使用阴影叠加出的立体文本特效代码如下，演示效果如图 4.26 所示。通过左上和右下各添加一个 1px 错位的补色阴影，营造出一种淡淡的立体效果。



Note

```
<style type="text/css">
body {background: #000;}
p {
    text-align: center;
    padding: 24px;
    margin: 0;
    font-family: helvetica, arial, sans-serif;
    font-size: 80px;
    font-weight: bold;
    color: #D1D1D1;
    background: #CCC;
    text-shadow: -1px -1px white,
        1px 1px #333;
}
</style>
<p>HTML5+CSS3</p>
```

【示例 6】反向思维，利用示例 5 的设计思路，也可以设计一种凹体效果，设计方法就是把示例 5 中左上和右下阴影颜色颠倒即可。主要代码如下，演示效果如图 4.27 所示。

```
<style type="text/css">
body {background: #000;}
p {
    text-align: center;
    padding: 24px;
    margin: 0;
    font-family: helvetica, arial, sans-serif;
    font-size: 80px;
    font-weight: bold;
    color: #D1D1D1;
    background: #CCC;
    text-shadow: 1px 1px white,
        -1px -1px #333;
}
</style>
<p>HTML5+CSS3</p>
```

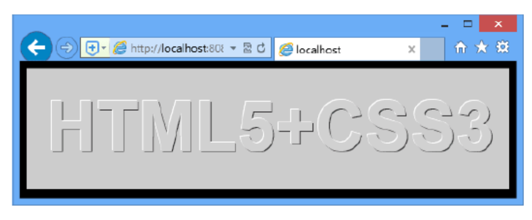


图 4.26　定义凸起的文字效果



图 4.27　定义凹下的文字效果

【示例 7】使用 text-shadow 属性可以为文本描边，设计方法是分别为文本 4 个边添加 1px 的实体阴影。代码如下，演示效果如图 4.28 所示。

```
<style type="text/css">
body {background: #000;}
p {
```



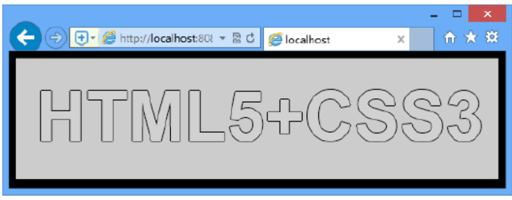



```
        text-align: center;
        padding:24px;
        margin:0;
        font-family: helvetica, arial, sans-serif;
        font-size: 80px;
        font-weight: bold;
        color: #D1D1D1;
        background:#CCC;
        text-shadow: -1px 0 black,
                0 1px black,
                1px 0 black,
                0 -1px black;
}
</style>
<p>HTML5+CSS3</p>
```

【示例 8】设计阴影不发生位移，同时定义阴影模糊显示，这样就可以模拟出文字外发光效果。代码如下，演示效果如图 4.29 所示。

```
<style type="text/css">
body {background: #000;}
p {
        text-align: center;
        padding:24px;
        margin:0;
        font-family: helvetica, arial, sans-serif;
        font-size: 80px;
        font-weight: bold;
        color: #D1D1D1;
        background:#CCC;
        text-shadow: 0 0 0.2em #F87,
                0 0 0.2em #F87;
}
</style>

<p>HTML5+CSS3</p>
```



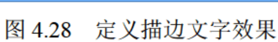


图 4.28　定义描边文字效果



图 4.29　定义外发光文字效果

4.4　内容生成和替换

content 属性属于内容生成和替换模块，可以为匹配的元素动态生成内容。这样就能够满足在 CSS



样式设计中临时添加非结构性的样式服务标签或者补充说明性内容等。

权威参考：http://www.w3.org/TR/css3-content/。



权威参考



视频讲解



4.4.1　定义 content

content 属性的简明语法如下：

```
content: normal | string | attr() | url() | counter() | none;
```

取值简单说明如下。

- ☑ normal：默认值，表现与 none 值相同。
- ☑ string：插入文本内容。
- ☑ attr()：插入元素的属性值。
- ☑ url()：插入一个外部资源，如图像、音频、视频或浏览器支持的其他任何资源。
- ☑ counter()：计数器，用于插入排序标识。
- ☑ none：无任何内容。

提示： content 属性早在 CSS2.1 中就被引入，可以使用:before 和:after 伪元素生成内容。此特性目前已被大部分的浏览器支持，另外 Opera 9.5+和 Safari 4 已经支持所有元素的 content 属性，而不仅仅是:before 和:after 伪元素。

在 CSS3 Generated Content 工作草案中，content 属性添加了更多的特征，如插入以及移除文档内容的能力，可以创建脚注、段落注释等。但目前还没有浏览器支持 content 的扩展功能。

【示例 1】 下面示例使用 content 属性为页面对象添加外部图像，演示效果如图 4.30 所示。

```
<style type="text/css">
div:after {
    border: solid 10px red;
    content: url(images/bg.png); /*在 div 元素内添加图片*/
}
</style>
<div>
 <h2>动态生成的图片</h2>
</div>
```

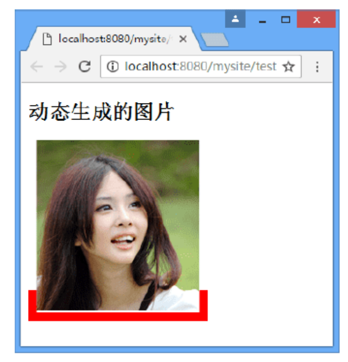


图 4.30　动态生成图像演示效果





注意：content 属性通常与:after 及:before 伪元素一起使用，在对象前或后显示内容。

【示例 2】 下面示例使用 content 属性把超链接的 URL 字符串动态显示在页面中，演示效果如图 4.31 所示。

```
<style type="text/css">
a:after {
    content: attr(href);
}
</style>
<a   href="http://www.baidu.com/">百度</a>
```

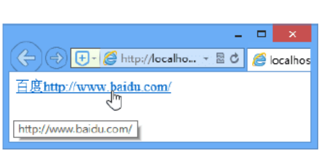


图 4.31　把属性值显示在页面中

4.4.2　案例：应用 content



下面结合多个示例练习 content 在网页中的应用。

【示例 1】 下面示例使用 content 属性，配合 CSS 计数器设计多层嵌套有序列表序号样式，效果如图 4.32 所示。

```
<style type="text/css">
ol {list-style:none;}                                  /*清除默认的序号*/
li:before {color:#f00; font-family:Times New Roman;}  /*设计层级目录序号的字体样式*/
li{counter-increment:a 1;}                             /*设计递增函数 a，递增起始值为 1*/
li:before{content:counter(a)". ";}                     /*把递增值添加到列表项前面*/
li li{counter-increment:b 1;}                          /*设计递增函数 b，递增起始值为 1*/
li li:before{content:counter(a)"."counter(b)". ";}     /*把递增值添加到二级列表项前面*/
li li li{counter-increment:c 1;}                       /*设计递增函数 c，递增起始值为 1*/
li li li:before{content:counter(a)"."counter(b)"."counter(c)". ";}  /*把递增值添加到三级列表项前面*/
</style>
<h1>网站导航</h1>
<ol>
    <li>新闻
        <ol>
            <li>国际新闻</li>
            <li>国内新闻
                <ol>
                    <li>互联网/科技</li>
                    <li>财经/理财</li>
                </ol>
            </li>
        </ol>
    </li>
    <li>交互</li>
</ol>
```





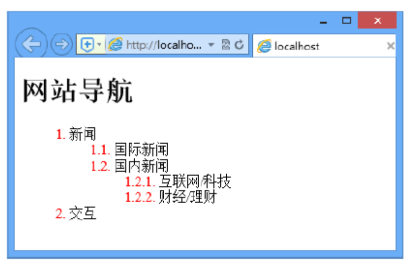


图 4.32　使用 CSS 技巧设计多级层级目录序号

【示例 2】下面示例使用 content 为引文动态添加引号，演示效果如图 4.33 所示。

```
<style type="text/css">
/*为不同语言指定引号的表现*/
:lang(en) > q {quotes:""" """;}
:lang(no) > q {quotes:"«" "»";}
:lang(ch) > q {quotes:"“" "”";}
/*在 q 标签的前后插入引号*/
q:before {content:open-quote;}
q:after   {content:close-quote;}
</style>
<p lang="no"><q>HTML5+CSS3 从入门到精通</q></p>
<p lang="en"><q>CSS Generated Content Module Level 3</p>
<p lang="ch"><q>CSS 生成内容模块 3.0</q></p>
```

【示例 3】下面示例使用 content 为超链接动态添加类型图标，演示效果如图 4.34 所示。

```
<style type="text/css">
a[href $=".pdf"]:after {
    content:url(images/icon_pdf.png);
}
a[rel = "external"]:after {
    content:url(images/icon_link.png);
}
</style>
<a href="http://www.book.com/1688.pdf">《HTML5+CSS3 从入门到精通》</a><br>
<a href="http://www.book.com/1688/" rel="external">《HTML5+CSS3 从入门到精通》</a>
```

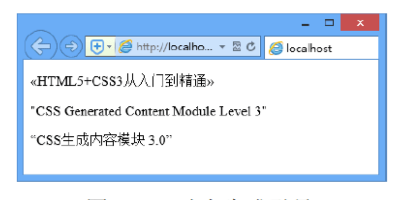


图 4.33　动态生成引号

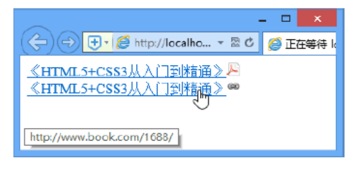


图 4.34　动态生成超链接类型图标

4.5　网 络 字 体

CSS3 允许用户通过@font-face 规则加载网络字体文件，方便用户自定义字体类型。@font-face







规则在 CSS3 规范中属于字体模块。

权威参考：http://www.w3.org/TR/css3-fonts/#font-face。

4.5.1　使用@font-face

@font-face 规则的语法格式如下：

```
@font-face {<font-description>}
```

@font-face 规则的选择符是固定的，用来引用网络字体文件。<font-description>是一个属性名值对，格式类似如下样式：

```
descriptor: value;
descriptor: value;
descriptor: value;
descriptor: value;
[...]
descriptor: value;
```

属性及其取值说明如下。

☑ font-family：设置文本的字体名称。
☑ font-style：设置文本样式。
☑ font-variant：设置文本是否大小写。
☑ font-weight：设置文本的粗细。
☑ font-stretch：设置文本是否横向拉伸变形。
☑ font-size：设置文本字体大小。
☑ src：设置自定义字体的相对或者绝对路径。注意，该属性只用在@font-face 规则里。

提示： 事实上，IE 5 已经开始支持该属性，但是只支持微软自有的.eot（Embedded Open Type）字体格式，而其他浏览器直到现在都不支持这一字体格式。不过，从 Safari 3.1 开始，用户可以设置.ttf（TrueType）和.otf（OpenType）两种字体作为自定义字体。考虑到浏览器的兼容性，在使用时建议同时定义.eot 和.ttf，以便能够兼容所有主流浏览器。

【示例】 下面是一个简单的示例，演示如何使用@font-face 规则在页面中使用网络字体。示例代码如下，演示效果如图 4.35 所示。

```
<style type="text/css">
/*引入外部字体文件*/
@font-face {
    /*选择默认的字体类型*/
    font-family: "lexograph";
    /*兼容 IE*/
    src: url(http://randsco.com//fonts/lexograph.eot);
    /*兼容非 IE*/
    src: local("Lexographer"), url(http://randsco.com/fonts/lexograph.ttf) format("truetype");
}
h1 {
    /*设置引入字体文件中的 lexograph 字体类型*/
    font-family: lexograph, verdana, sans-serif;
```



```
        font-size:4em;}
</style>
<h1>http://www.baidu.com/</h1>
```



Note



图 4.35　设置为 lexograph 字体类型的文字

提示：嵌入外部字体需要考虑用户带宽问题，因为一个中文字体文件小的有几个兆，大的有十几兆，这么大的字体文件下载过程会出现延迟，同时服务器也不能忍受如此频繁的申请下载。如果只是想标题使用特殊字体，最好设计成图片。



视频讲解

4.5.2　案例：设计字体图标

本节示例通过@font-face 规则引入外部字体文件 glyphicons-halflings-regular.eot，然后定义几个字体图标，嵌入导航菜单项目中，效果如图 4.36 所示。



图 4.36　设计包含字体图标的导航菜单

示例主要代码如下：

```
<style type="text/css">
/*引入外部字体文件*/
@font-face {
    font-family: 'Glyphicons Halflings';            /*选择默认的字体类型*/
    /*外部字体文件列表*/
    src: url('fonts/glyphicons-halflings-regular.eot');
    src: url('fonts/glyphicons-halflings-regular.eot?#iefix') format('embedded-opentype'),
        url('fonts/glyphicons-halflings-regular.woff2') format('woff2'),
        url('fonts/glyphicons-halflings-regular.woff') format('woff'),
        url('fonts/glyphicons-halflings-regular.ttf') format('truetype'),
        url('fonts/glyphicons-halflings-regular.svg#glyphicons_halflingsregular') format('svg');
}
/*定义字体图标样式*/
.glyphicon {
    position: relative;                             /*相对定位*/
    top: 1px;                                       /*相对向上偏移 1px*/
    display: inline-block;                          /*行内块显示*/
    font-family: 'Glyphicons Halflings';            /*定义字体类型*/
```



Note

```
        font-style: normal;                          /*字体样式*/
        font-weight: normal;                         /*字体粗细*/
        line-height: 1;                              /*定义行高，清除文本行对图标的影响*/
        -webkit-font-smoothing: antialiased;         /*兼容谷歌浏览器解析*/
        -moz-osx-font-smoothing: grayscale;          /*兼容 Firefox 浏览器解析*/
}
.glyphicon-home:before {content: "\e021";}
.glyphicon-user:before {content: "\e008";}
.glyphicon-search:before {content: "\e003";}
.glyphicon-plus:before {content: "\e081";}
span {/*定义字体图标标签样式*/
        font-size: 16px;
        color: red;
}
ul {/*定义导航列表框样式，清除默认样式*/
        margin: 0;
        padding: 0;
        list-style: none;
}
li {/*定义列表项目样式，水平并列显示*/
        float: left;
        padding: 6px 12px;
        margin: 3px;
        border: solid 1px hsla(359,93%,69%,0.6);
        border-radius: 6px;
}
li a {/*定义超链接文本样式*/
        font-size: 16px;
        color: red;
        text-decoration: none;
}
</style>
<ul>
        <li><span class="glyphicon glyphicon-home"></span> <a href="#">主页</a></li>
        <li><span class="glyphicon glyphicon-user"></span> <a href="#">登录</a></li>
        <li><span class="glyphicon glyphicon-search"></span> <a href="#">搜索</a></li>
        <li><span class="glyphicon glyphicon-plus"></span> <a href="#">添加</a></li>
</ul>
```

4.6　案 例 实 战

本节将以案例形式实战练习 CSS3 新增的文本属性。

4.6.1　设计黑科技网站首页



视 频 讲 解

本案例将模拟一个黑科技网站的首页，借助 text-shadow 属性设计阴影效果，通过颜色的搭配，营造一种静谧而又神秘的画面，使用两幅 PNG 图像对页面效果进行装饰和点缀，最后演示效果如图 4.37



所示。







图 4.37　设计黑科技网站首页

案例主要代码如下：

```
<style type="text/css">
body {/*页面样式*/
    padding: 0px; margin: 0px;    /*清除页边距*/
    background: black; color: #666;}
#text-shadow-box {/*设计包含框样式*/
    /*定义内部的定位元素以这个框为参照物*/
    position: relative;
    width: 598px; height: 406px;
    background: #666;
    /*禁止内容超过设定的区域*/
    overflow: hidden;
    border: #333 1px solid;}
#text-shadow-box div.wall {/*设计背景墙样式*/
    position: absolute;
    width: 100%;
    top: 175px; left: 0px}
#text {/*设计导航文本样式*/
    text-align: center; line-height: 0.5em;
    margin: 0px;
    font-family: helvetica, arial, sans-serif;
    height: 1px;
    color: #999; font-size: 80px; font-weight: bold;
    text-shadow: 5px -5px 16px #000;}
div.wall div {/*设计遮盖层样式*/
    position: absolute;
    width: 100%; height: 300px;
    top: 42px; left: 0px;
    background: #999;}
/*设计覆盖在上面的探照灯效果图*/
#spotlight {
    position: absolute;
```



```
        width: 100%; height: 100%;
        top: 0px; left: 0px;
        background: url(images/spotlight.png) center -300px;
        font-size: 12px;}
    #spotlight a {
        color: #ccc; text-decoration: none;
        position: absolute;
        left: 45%; top: 58%;
        text-shadow: 1px 1px #999, -1px -1px #333;}
    #cat {
        position: absolute;
        top: 130px; left: 260px;
        z-index: 1000;
        opacity: 0.5;}
    #cat img {width: 80px;}
    </style>
    <div id="text-shadow-box">
        <div class="wall">
            <p id="text">CSS3 黑科技</p>
            <div></div>
        </div>
        <div id="spotlight"><a href="index.htm">入口</a></div>
        <div id="cat"><img src="images/cat.png"></div>
    </div>
```



定义页面背景色为黑色，前景色为灰色。设计右上偏移的阴影，适当进行模糊处理，产生色晕效果，阴影色为深色，营造静谧的主观效果。设计一个遮罩层，让其覆盖在页面上，使其满窗口显示，通过前期设计好的一个探照灯背景来营造神秘效果。通过<div id="spotlight">外罩，为页面覆盖一层桌纸，添加特殊的艺术效果。

4.6.2　设计消息提示框



本节将借助 CSS3 增强的文本特性和相关动画功能，设计一个纯 CSS 的消息提示框，效果如图 4.38 所示。



图 4.38　设计消息提示框





Note

【操作步骤】

第 1 步，设计消息框基本框架样式。

```
.bubble {
    width: 200px; eight: 50px;              /*定义消息框大小，可忽略，让消息框自由收缩*/
    background:hsla(93,96%,62%,1);          /*定义背景色，必须与下面箭头背景色保持一致*/
    padding: 12px;                          /*增加补白，防止消息文本跑到框外*/
    position: relative;                     /*定义定位包含框，方便定位箭头*/
    border-radius: 8px;                     /*圆角*/
}
```

第 2 步，以内容生成的方式，设计箭头基本样式。

```
.bubble:before {
    content: "";                            /*不显示任何内容*/
    width: 0; height: 0;                    /*定义箭头内容区大小*/
    position: absolute;                     /*绝对定位*/
    z-index:-1;                             /*显示在消息框的下面*/
}
```

第 3 步，设计左侧消息提示框的扩展样式。

```
.bubble.bubble-left:before {
    /*调整箭头的位置，right 值不动，top 值可以微调，百分比值，能自适应消息框的大小变化*/
    right: 90%;
    top: 50%;
    /*定义箭头的倾斜角度*/
    transform: rotate(-25deg);
    /*定义箭头的长短、粗细。border-top 和 border-bottom 控制粗糙和偏向，
    border-right 控制长短，其颜色值必须与消息框背景色保持一致*/
    border-top: 20px  solid transparent;
    border-right: 80px  solid  hsla(93,96%,62%,1);
    border-bottom: 20px  solid transparent;
}
```

第 4 步，模仿第 3 步，设计右侧消息提示框的扩展样式。

```
.bubble.bubble-right:before {
    left: 90%; top: 50%;
    transform: rotate(25deg);
    border-top: 20px  solid transparent;
    border-left: 80px  solid hsla(93,96%,62%,1);
    border-bottom: 20px  solid transparent;
}
```

第 5 步，模仿第 3 步，设计顶部消息提示框的扩展样式。

```
.bubble.bubble-top:before {
    bottom: 80%; left: 50%;
    transform: rotate(25deg);
    border-left: 20px  solid transparent;
    border-bottom: 80px  solid hsla(93,96%,62%,1);
    border-right: 20px  solid transparent;
}
```





第 6 步，设计底部消息提示框的扩展样式。

```
.bubble.bubble-bottom:before {
    top: 80%;left: 50%;
    transform: rotate(25deg);
    border-left: 20px   solid transparent;
    border-top: 80px   solid hsla(93,96%,62%,1);
    border-right: 20px   solid transparent;
}
```

第 7 步，在文档中插入消息提示框，使用类样式进行定义。bubble 定义消息框，bubble-left、bubble-right、bubble-bottom 和 bubble-top 定义不同方向显示。代码如下：

```
<div class="bubble bubble-left">左侧消息提示框<br>class="bubble bubble-left"</div>
<div class="bubble bubble-right">右侧消息提示框<br>class="bubble bubble-right"</div>
<div class="bubble bubble-bottom">底部消息提示框<br>class="bubble bubble-bottom"</div>
<div class="bubble bubble-top">顶部消息提示框<br>class="bubble bubble-top"</div>
```

4.7　在线练习

本节将通过大量的上机示例，帮助初学者练习使用 HTML5 语义标签灵活定义网页文本样式和版式：（1）标识文本；（2）文本流方向。


在线练习 1


在线练习 2



第5章 使用 CSS 设计背景样式

在 CSS2.1 中，background 属性的功能还无法满足设计的需求，为了方便设计师更灵活地设计需要的网页效果，CSS3 在原有 background 基础上新增了一些功能属性，可以在同一个对象内叠加多个背景图像，可以改变背景图像的尺寸，还可以指定背景图像的显示范围和绘制起点等。另外，CSS3 允许用户使用渐变函数绘制背景图像，这极大地降低了网页设计的难度，激发了设计师的创意灵感。

【学习重点】

- 设置背景图像的原点、大小。
- 正确使用背景图像裁切属性。
- 灵活使用多重背景图像设计网页版面。
- 正确使用线性渐变和径向渐变。
- 熟练使用渐变函数设计网页元件。



Note

5.1　设计背景图像

CSS3 增强了 background 属性的功能，允许在同一个元素内叠加多个背景图像，还新增了 3 个与背景相关的属性：background-clip、background-origin、background-size。下面分别进行介绍。

权威参考：http://www.w3.org/TR/css3-background/。



权 威 参 考



视 频 讲 解

5.1.1　设置背景图像

在 CSS 中可以使用 background-image 属性来定义背景图像。具体用法如下：

```
background-image: none | <url>
```

默认值为 none，表示无背景图；<url>表示使用绝对或相对地址指定背景图像。

> 提示：GIF 格式图像可以设计动画、透明背景，具有图像小巧等优点，而 JPG 格式图像具有更丰富的颜色数，图像品质相对要好，PNG 类型则综合了 GIF 和 JPG 两种格式图像的优点。

【示例】如果背景包含透明区域的 GIF 或 PNG 格式图像，则被设置为背景图像时，这些透明区域依然被保留。在下面这个示例中，先为网页定义背景图像，然后为段落文本定义透明的 GIF 背景图像，显示效果如图 5.1 所示。

```
<style type="text/css">
html, body, p{height:100%;}
body {background-image:url(images/bg.jpg);}
p {background-image:url(images/ren.png);}
</style>
<p></p>
```



图 5.1　透明背景图像的显示效果


视频讲解


Note

5.1.2 设置显示方式

CSS 使用 background-repeat 属性控制背景图像的显示方式。具体用法如下：

```
background-repeat: repeat-x | repeat-y | [repeat | space | round | no-repeat]{1,2}
```

取值说明如下。

☑ repeat-x：背景图像在横向上平铺。
☑ repeat-y：背景图像在纵向上平铺。
☑ repeat：背景图像在横向和纵向平铺。
☑ space：背景图像以相同的间距平铺且填充满整个容器或某个方向，仅 CSS3 支持。
☑ round：背景图像自动缩放直到适应且填充满整个容器，仅 CSS3 支持。
☑ no-repeat：背景图像不平铺。

【示例】下面示例设计一个公司公告栏，其中宽度是固定的，但是高度可能会根据正文内容进行动态调整，为了适应这种设计需要，不妨利用垂直平铺来进行设计。

【操作步骤】

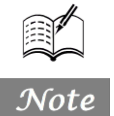
第 1 步，把“公司公告”栏目分隔为上、中、下 3 块，设计上和下为固定宽度，而中间块为可以随时调整高度。设计的结构如下：

```
<div id="call">
    <div id="call_tit">公司公告</div >
    <div id="call_mid"></div >
    <div id="call_btm"></div >
</div>
```

第 2 步，所实现的样式表如下，最后经过调整中间块元素的高度以形成不同高度的公告牌，演示效果如图 5.2 所示。

```
<style type="text/css">
#call {
    width:218px;                                    /*固定宽度*/
    font-size:14px;                                 /*字体大小*/
}
#call_tit {
    background:url(images/call_top.gif);            /*头部背景图像*/
    background-repeat:no-repeat;                    /*不平铺显示*/
    height:43px;                                    /*固定高度，与背景图像高度一致*/
    color:#fff;                                     /*白色标题*/
    font-weight:bold;                               /*粗体*/
    text-align:center;                              /*居中显示*/
    line-height:43px;                               /*标题垂直居中*/
}
#call_mid {
    background-image:url(images/call_mid.gif);      /*背景图像*/
    background-repeat:repeat-y;                     /*垂直平铺*/
    height:160px;                                   /*可自由设置的高度*/
}
#call_btm {
    background-image:url(images/call_btm.gif);      /*底部背景图像*/
    background-repeat:no-repeat;                    /*不平铺显示*/
```





```
        height:11px;                               /*固定高度，与背景图像高度一致*/
    }
```

图 5.2　背景图像垂直平铺示例模拟效果

5.1.3　设置显示位置

在默认情况下，背景图像显示在元素的左上角，并根据不同方式执行不同显示效果。为了更好地控制背景图像的显示位置，CSS 定义了 background-position 属性来精确定位背景图像。

background-position 属性取值包括两个值，分别用来定位背景图像的 x 轴、y 轴坐标，取值单位没有限制。具体用法如下：

```
    background-position: [left | center | right | top | bottom | <percentage> | <length>] | [left | center | right | <percentage> | <length> ] [top | center | bottom | <percentage> | <length>] | [ center | [ left | right] [<percentage> | <length>]? ] && [ center | [top | bottom] [<percentage> | <length>]?]
```

默认值为 0% 0%，等效于 left top。

【示例】下面示例利用 4 个背景图像拼接起来组成一个栏目版块。这些背景图像分别被定位到栏目的 4 个边上，形成一个圆角的矩形，并富有立体感，效果如图 5.3 所示。



图 5.3　背景图像定位综合应用





Note

实例所用到的 HTML 结构代码如下：

```
<div id="explanation">
    <h3><span>这是什么？</span></h3>
    <p class="p1"><span><span class="first">对</span>于网页设计师来说应该好好研究<acronym title="cascading style sheets">CSS</acronym>。Zen Garden 致力于推广和使用 CSS 技术，努力激发和鼓励您的灵感和参与。读者可以从浏览高手的设计作品入门。只要选择列表中的任一个样式表，就可以将它加载到这个页面中。<acronym title="hypertext markup language">HTML</acronym>文档结构始终不变，但是读者可以自由地修改和定义<acronym title="cascading style sheets">CSS</acronym>样式表。</span></p>
    <p class="p2"><span><acronym title="cascading style sheets">CSS</acronym>具有强大的功能，可以自由控制 HTML 结构。当然读者需要拥有驾驭 CSS 技术的能力和创意的灵感，同时亲自动手，用具体的实例展示 CSS 的魅力，展示个人的才华。截至目前，很多 Web 设计师和程序员已经介绍过关于 CSS 应用技巧和兼容技术的各种技巧和案例。而平面设计师还没有足够重视 CSS 的潜力。读者是不是需要从现在开始呢？</span></p>
</div>
```

根据这个 HTML 结构所设计的 CSS 样式表如下（请注意背景图像的定位方法）：

```
<STYLE type="text/css">
body { /*定义网页背景色、居中显示、字体颜色*/
    background:#DFDFDF; text-align:center; color:#454545;
}
p, h3 {margin:0; padding:0;}/*清除段落和标题的默认边距*/
#explanation {
    background-color:#ffffff;                          /*白色背景，填充所有区域*/
    background-image:url(images/img_explanation.jpg);  /*指定背景图像*/
    background-position:left bottom;                   /*定位背景图像位于左下角*/
    background-repeat:repeat-y;                        /*在垂直方向上平铺背景图像*/
    width:546px;                                       /*固定栏目宽度*/
    margin:0 auto;                                     /*栏目居中显示*/
    font-size:13px; line-height:1.6em; text-indent:2em; /*定义栏目内字体属性*/
}
#explanation h3 {
    background:url(images/title_explanation.gif) no-repeat; /*顶部背景图像，不平铺*/
    height:39px;                                       /*固定标题栏高度*/
}
#explanation h3 span { display:none; }                 /*隐藏标题栏内信息*/
#explanation p {                                       /*定义右侧背景图像，垂直平铺*/
   background:url(images/right_bg.gif) right repeat-y;}
#explanation .p2 span {                                /*底部背景图像，不平铺*/
    padding-bottom:20px;                               /*增加第 2 段底部内边距，显示背景图像*/
    background:url(images/right_bottom.gif) bottom no-repeat;
}
#explanation p span {/*定义段落文本左侧的内边距，以便显示左侧背景图像*/
    padding:0 15px 10px 77px;
    display:block;                                     /*定义块状显示，内边距才有效*/
    text-align:left;                                   /*文本左对齐*/
}
#explanation p .first {                                /*定义首字下沉特效*/
    font-size:60px; color:#820015; line-height:1em;    /*字体显示属性*/
    float:left;                                        /*向左浮动*/
    padding:0;                                         /*清除上面样式为段落定义的内边距*/
```



```
}
</STYLE>
```

在上面的样式表中，分别为不同元素定义背景图像，然后通过定位技术把背景图像定位到对应的 4 个边上，并根据需要运用平铺技术实现圆角区域效果。



注意： 百分比是最灵活的定位方式，同时也是最难把握的定位单位。

在默认状态下，定位的位置为（0% 0%），定位点是背景图像的左上顶点，定位距离是该点到包含框左上角顶点的距离，即两点重合。

如果定位背景图像位置为（100% 100%），则定位点是背景图像的右下顶点，定位距离是该点到包含框左上角顶点的距离，这个距离等于包含框的宽度和高度。

百分比也可以取负值，负值的定位点是包含框的左上顶点，而定位距离则由图像自身的宽和高来决定。

CSS 还提供了 5 个关键字：left、right、center、top 和 bottom。这些关键字实际上就是百分比特殊值的一种固定用法。详细列表说明如下：

```
/*普通用法*/
top left、left top                      = 0% 0%
right top、top right                    = 100% 0%
bottom left、left bottom                = 0% 100%
bottom right、right bottom              = 100% 100%
/*居中用法*/
center、center center                   = 50% 50%
/*特殊用法*/
top、top center、center top             = 50% 0%
left、left center、center left          = 0% 50%
right、right center、center right       =100% 50%
bottom、bottom center、center bottom    = 50% 100%
```

5.1.4　设置固定背景



视频讲解

在默认情况下，背景图像能够跟随网页内容上下滚动。可以使用 background-attachment 属性定义背景图像在窗口内固定显示，具体用法如下：

```
background-attachment: fixed | local | scroll
```

默认值为 scroll，具体取值说明如下。

☑ fixed：背景图像相对于浏览器窗体固定。

☑ scroll：背景图像相对于元素固定，也就是说当元素内容滚动时背景图像不会跟着滚动，因为背景图像总是要跟着元素本身。

☑ local：背景图像相对于元素内容固定，也就是说当元素内容滚动时背景图像也会跟着滚动，此时不管元素本身是否滚动，当元素显示滚动条时才会看到效果。该属性值仅 CSS3 支持。

【示例】 在下面的示例中，为<body>标签设置背景图片，且不平铺、固定，这时通过拖动浏览器滚动条，可以看到网页内容在滚动，而背景图片静止显示。页面演示效果如图 5.4 所示。

```
<style type="text/css">
body {
```



```
        background-image: url(images/bg.jpg);          /*设置背景图片*/
        background-repeat: no-repeat;                  /*背景图片不平铺*/
        background-position: left center;              /*背景图片的位置*/
        background-attachment: fixed;                  /*背景图片固定，不随滚动条滚动而滚动*/
        height: 1200px;                                /*高度，出现浏览器的滚动条*/
    }
    #box {float:right; width:400px;}
    </style>
    <div id="box">
        <h1> 雨巷</h1>
        <h2>戴望舒</h2>
        <pre>
    撑着油纸伞，独自
    彷徨在悠长、悠长
    又寂寥的雨巷，
    我希望逢着
    一个丁香一样的
    结着愁怨的姑娘。
    ……
        </pre>
    </div>
```

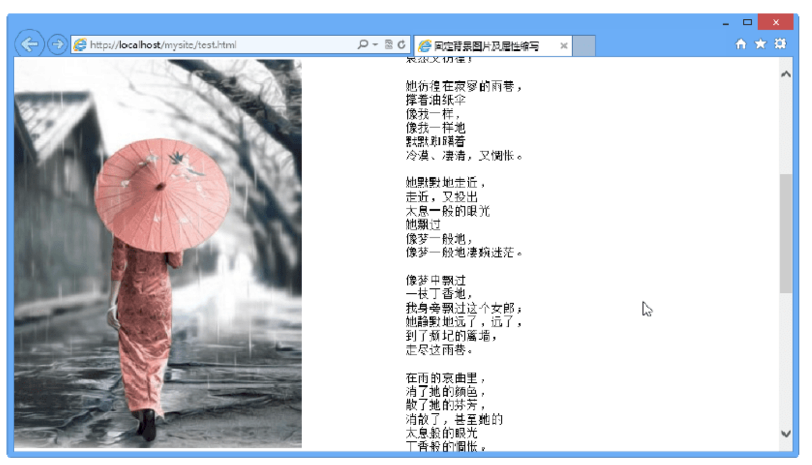


图 5.4　背景图片固定



视 频 讲 解

5.1.5　设置定位原点

background-origin 属性定义 background-position 属性的定位原点。在默认情况下，background-position 属性总是以元素左上角为坐标原点定位背景图像，使用 background-origin 属性可以改变这种定位方式。该属性的基本语法如下：

```
background-origin: border-box | padding-box | content-box;
```

取值简单说明如下。

☑ border-box：从边框区域开始显示背景。

☑ padding-box：从补白区域开始显示背景，为默认值。



☑　content-box：仅在内容区域显示背景。

【示例】background-origin 属性改善了背景图像定位的方式，更灵活地决定背景图像应该显示的位置。下面示例利用 background-origin 属性重设背景图像的定位坐标，以便更好地控制背景图像的显示，演示效果如图 5.5 所示。



图 5.5　设计诗词效果

示例代码如下：

```
<style type="text/css">
div {/*定义包含框的样式*/
    height: 322px;
    width: 780px;
    border: solid 1px red;
    padding: 250px 4em 0;
    /*为了避免背景图像重复平铺到边框区域，应禁止它平铺*/
    background:url(images/p3.jpg) no-repeat;
    /*设计背景图像的定位坐标点为元素边框的左上角*/
    background-origin:border-box;
    /*将背景图像等比缩放到完全覆盖包含框，背景图像有可能超出包含框*/
    background-size:cover;
    overflow:hidden;                    /*隐藏超出包含框的内容*/
}
div h1, div h2{                         /*定义标题样式*/
    font-size:18px; font-family:"幼圆";
    text-align:center;                  /*水平居中显示*/
}
div p {/*定义正文样式*/
    text-indent:2em;                    /*首行缩进 2 个字符*/
    line-height:2em;                    /*增大行高，让正文看起来更疏朗*/
    margin-bottom:2em;                  /*调整底部边界，增大段落文本距离*/
}
```

Note



```
</style>
<div>
    <h1>念奴娇&#8226;赤壁怀古</h1>
    <h2>苏轼</h2>
    <p>大江东去，浪淘尽，千古风流人物。故垒西边，人道是，三国周郎赤壁。乱石穿空，惊涛拍岸，卷起千堆雪。江山如画，一时多少豪杰。</p>
    <p>遥想公瑾当年，小乔初嫁了，雄姿英发。羽扇纶巾，谈笑间，樯橹灰飞烟灭。故国神游，多情应笑我，早生华发。人生如梦，一尊还酹江月。</p>
</div>
```



Note



视频讲解

5.1.6 设置裁剪区域

background-clip 属性定义背景图像的裁剪区域。该属性的基本语法如下：

```
background-clip:border-box | padding-box | content-box | text;
```

取值简单说明如下。

☑ border-box：从边框区域向外裁剪背景，为默认值。

☑ padding-box：从补白区域向外裁剪背景。

☑ content-box：从内容区域向外裁剪背景。

☑ text：从前景内容（如文字）区域向外裁剪背景。

提示：如果取值为 padding-box，则 background-image 将忽略补白边缘，此时边框区域显示为透明。

如果取值为 border-box，则 background-image 将包括边框区域。

如果取值为 content-box，则 background-image 将只包含内容区域。

如果 background-image 属性定义了多重背景，则 background-clip 属性值可以设置多个值，并用逗号分隔。

如果 background-clip 属性值为 padding-box，background-origin 属性值为 border-box，且 background-position 属性值为 top left（默认初始值），则背景图左上角将会被截取掉部分。

【示例 1】下面示例演示如何设计背景图像仅在内容区域内显示，演示效果如图 5.6 所示。

```
<style type="text/css">
div {
    height:150px;
    width:300px;
    border:solid 50px gray;
    padding:50px;
    background:url(images/bg.jpg) no-repeat;
    /*将背景图像等比缩放到完全覆盖包含框，背景图像有可能超出包含框*/
    background-size:cover;
    /*将背景图像从 content 区域开始向外裁剪背景*/
    background-clip:content-box;
}
</style>

<div></div>
```



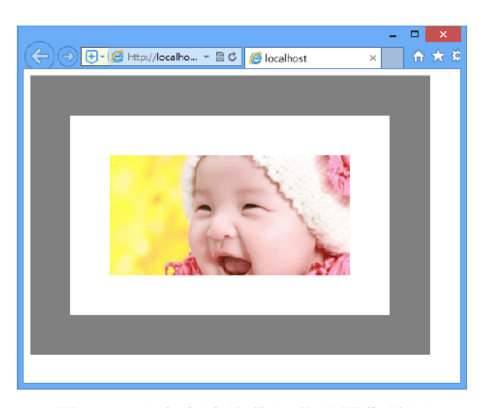


图 5.6　以内容边缘裁切背景图像效果



【示例 2】下面示例同时定义 background-clip 和 background-origin 属性值为 content，可以设计比较特殊的按钮样式，演示效果如图 5.7 所示。

```
<style type="text/css">
button {
    height:40px;                        /*固定包含框大小*/
    width:150px;
    padding:1px;                        /*在内容区留点空隙*/
    cursor:pointer;                     /*定义手形指针样式*/
    color:#fff;                         /*白色字体*/
    /*设计立体边框样式*/
    border:3px double #95071b;
    border-right-color:#650513;
    border-bottom-color:#650513;
    /*为了避免背景图像重复平铺到边框区域，应禁止它平铺*/
    background:url(images/img6.jpg) no-repeat;
    /*设计背景图像的定位坐标点为元素内容区域的左上角*/
    background-origin:content-box;
    /*设计背景图像以内容区域的边缘裁切背景图像*/
    background-clip:content-box;
}
</style>
<button>导航按钮 >></button>
```

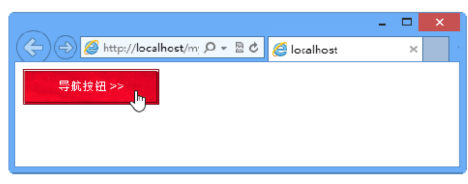


图 5.7　设计按钮效果

5.1.7　设置背景图像大小



background-size 可以控制背景图像的显示大小。该属性的基本语法如下：

```
background-size: [<length> | <percentage> | auto]{1,2} | cover | contain;
```



Note

取值简单说明如下。

☑ <length>：由浮点数字和单位标识符组成的长度值。不可为负值。

☑ <percentage>：取值范围为 0～100%。不可为负值。

☑ cover：保持背景图像本身的宽高比例，将图片缩放到正好完全覆盖所定义背景的区域。

☑ contain：保持图像本身的宽高比例，将图片缩放到宽度或高度正好适应所定义背景区域。

初始值为 auto。background-size 属性可以设置一个或两个值，一个为必填，一个为可选。其中第 1 个值用于指定背景图像的 width，第 2 个值用于指定背景图像的 height，如果只设置一个值，则第 2 个值默认为 auto。

【示例】下面示例使用 background-size 属性自由定制背景图像的大小，让背景图像自适应盒子的大小，从而设计与模块大小完全适应的背景图像，效果如图 5.8 所示，只要背景图像长宽比与元素长宽比相同，就不用担心背景图像变形显示。

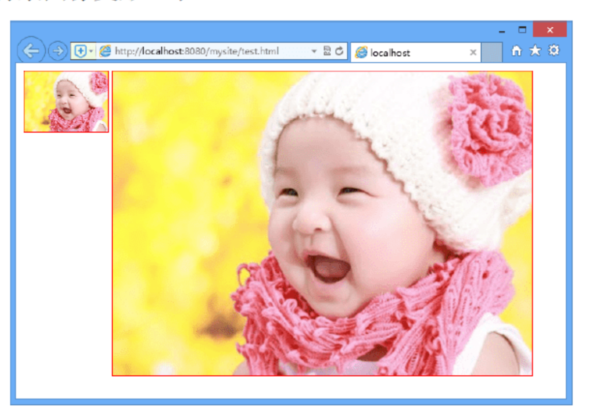


图 5.8　设计背景图像自适应显示

示例代码如下：

```
<style type="text/css">
div {
      margin:2px;
      float:left;
      border:solid 1px red;
      background:url(images/img2.jpg) no-repeat center;
      /*设计背景图像完全覆盖元素区域*/
      background-size:cover;}
/*设计元素大小*/
.h1 {height:80px; width:110px;}
.h2 {height:400px; width:550px;}
</style>
<div class="h1"></div>
<div class="h2"></div>
```



视 频 讲 解

5.1.8　设置多重背景图像

CSS3 支持在同一个元素内定义多个背景图像，还可以将多个背景图像进行叠加显示，从而使得设计多图背景栏目变得更加容易。





【示例 1】本例使用 CSS3 多背景设计花边框，使用 background-origin 属性定义仅在内容区域显示背景，使用 background-clip 属性定义背景从边框区域向外裁剪，如图 5.9 所示。

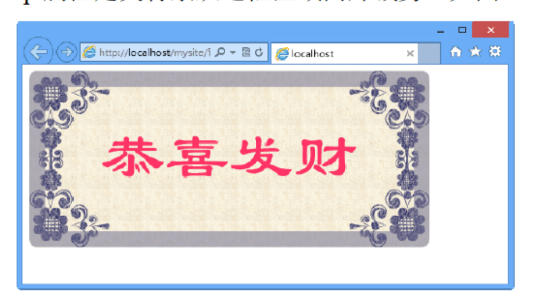


图 5.9　设计花边框效果

示例代码如下：

```
<style type="text/css">
.demo {
    /*设计元素大小、补白、边框样式，边框为 20px，颜色与背景图像色相同*/
    width: 400px; padding: 30px 30px; border: 20px solid rgba(104, 104, 142,0.5);
    /*定义圆角显示*/
    border-radius: 10px;
    /*定义字体显示样式*/
    color: #f36; font-size: 80px; font-family:"隶书";line-height: 1.5; text-align: center;
}
.multipleBg {
    /*定义 5 个背景图，分别定位到 4 个顶角，其中前 4 个禁止平铺，最后一个可以平铺*/
    background: url("images/bg-tl.png") no-repeat left top,
                url("images/bg-tr.png") no-repeat right top,
                url("images/bg-bl.png") no-repeat left bottom,
                url("images/bg-br.png") no-repeat right bottom,
                url("images/bg-repeat.png") repeat left top;
    /*改变背景图像的 position 原点，4 朵花都是 border 原点，而平铺背景是 padding 原点*/
    background-origin: border-box, border-box, border-box, border-box, padding-box;
    /*控制背景图像的显示区域，所有背景图像超过 border 外边缘都将被剪切掉*/
    background-clip: border-box;
}
</style>
<div class="demo multipleBg">恭喜发财</div>
```

【示例 2】在下面示例中利用 CSS3 多背景图功能设计圆角栏目，效果如图 5.10 所示。

```
<style type="text/css">
.roundbox {
    padding: 2em;
    /*为容器定义 8 个背景图像*/
    background-image: url(images/roundbox1/tl.gif),
                      url(images/roundbox1/tr.gif),
                      url(images/roundbox1/bl.gif),
                      url(images/roundbox1/br.gif),
                      url(images/roundbox1/right.gif),
                      url(images/roundbox1/left.gif),
```



Note

```
                                  url(images/roundbox1/top.gif),
                                  url(images/roundbox1/bottom.gif);
        /*定义 4 个顶角图像禁止平铺，4 个边框图像分别沿 x 轴或 y 轴平铺*/
        background-repeat: no-repeat,
                                  no-repeat,
                                  no-repeat,
                                  no-repeat,
                                  repeat-y,
                                  repeat-y,
                                  repeat-x,
                                  repeat-x;
        /*定义 4 个顶角图像分别固定在 4 个顶角位置，4 个边框图像分别固定在四边位置*/
        background-position: left 0px,
                                    right 0px,
                                    left bottom,
                                    right bottom,
                                    right 0px,
                                    0px 0px,
                                    left 0px,
                                    left bottom;
        background-color: #66CC33;
    }
    </style>
    <div class="roundbox">
        <h1>念奴娇&#8226;赤壁怀古</h1>
        <h2>苏轼</h2>
        <p>大江东去，浪淘尽，千古风流人物。故垒西边，人道是，三国周郎赤壁。乱石穿空，惊涛拍岸，卷起千堆雪。江山如画，一时多少豪杰。</p>
        <p>遥想公瑾当年，小乔初嫁了，雄姿英发。羽扇纶巾，谈笑间，樯橹灰飞烟灭。故国神游，多情应笑我，早生华发。人生如梦，一尊还酹江月。</p>
    </div>
```



图 5.10　定义多背景图像

注意： 每幅背景图像的源、定位坐标以及平铺方式的先后顺序要一一对应。

提示： 上面示例用到了 background-image、background-repeat 和 background-position 多个背景属性。这些属性都是 CSS1 中就有的属性，但是在 CSS3 中，允许同时指定多个属性值，多个属性值以逗号作为分隔符，用来指定多个背景图像的显示性质。



5.2　设计渐变背景

W3C 于 2010 年 11 月正式支持渐变背景样式，该草案作为图像值和图像替换内容模块的一部分进行发布，主要包括 linear-gradient()、radial-gradient()、repeating-linear-gradient()和 repeating-radial-gradient() 4 个渐变函数。

权威参考：http://dev.w3.org/csswg/css3-images/#gradients。



权 威 参 考



视 频 讲 解

5.2.1　定义线性渐变

创建一个线性渐变至少需要两个颜色，也可以选择设置一个起点或一个方向。简明语法格式如下：

```
linear-gradient(angle, color-stop1, color-stop2, …)
```

参数简单说明如下。

☑　angle：用来指定渐变的方向，可以使用角度或者关键字来设置。4 个关键字说明如下。

- ➢　to left：设置渐变从右到左，相当于 270deg。
- ➢　to right：设置渐变从左到右，相当于 90deg
- ➢　to top：设置渐变从下到上，相当于 0deg
- ➢　to bottom：设置渐变从上到下，相当于 180deg。该值为默认值。

提示： 如果创建对角线渐变，可以使用 to top left（从右下到左上）类似组合来实现。

☑　color-stop：用于指定渐变的色点，包括一个颜色值和一个起点位置，颜色值和起点位置以空格分隔。起点位置可以为一个具体的长度值（不可为负值），也可以是一个百分比值，如果是百分比值则参考应用渐变对象的尺寸，最终会被转换为具体的长度值。

【示例 1】 下面示例为<div id="demo">对象应用了一个简单的线性渐变背景，方向从上到下，颜色由白色到浅灰渐变显示，效果如图 5.11 所示。

```
<style type="text/css">
#demo {
      width:300px;
      height:200px;
      background: linear-gradient(#fff, #333);
}
</style>
<div id="demo"></div>
```

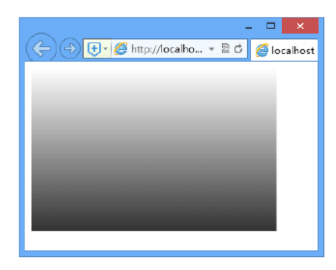


图 5.11　应用简单的线性渐变效果



Note

提示：针对示例 1，用户可以继续尝试做下面练习，实现不同的设置，得到相同的设计效果。

☑ 设置一个方向：从上到下，覆盖默认值。

linear-gradient(to bottom, #fff, #333);

☑ 设置反向渐变：从下到上，同时调整起止颜色位置。

linear-gradient(to top, #333, #fff);

☑ 使用角度值设置方向。

linear-gradient(180deg, #fff, #333);

☑ 明确起止颜色的具体位置，覆盖默认值。

linear-gradient(to bottom, #fff 0%, #333 100%);

【拓展】

最新主流浏览器都支持线性渐变的标准用法，但是考虑到安全性，用户应酌情兼容旧版本浏览器的私有属性。

Webkit 是第一个支持渐变的浏览器引擎（Safari 4+），它使用-webkit-gradient()私有函数支持线性渐变样式，简明用法如下：

```
-webkit-gradient(linear, point, point, stop)
```

参数简单说明如下。

☑ linear：定义渐变类型为线性渐变。

☑ point：定义渐变起始点和结束点坐标。该参数支持数值、百分比值和关键字，如(0 0)或者(left top)等。关键字包括 top、bottom、left 和 right。

☑ stop：定义渐变色和步长。包括 3 个值，即开始的颜色，使用 from(colorvalue)函数定义；结束的颜色，使用 to(colorvalue)函数定义；颜色步长，使用 color-stop(value, color value)函数定义。color-stop()函数包含两个参数值，第一个参数值为一个数值或者百分比值，取值范围为 0～1.0（或者 0%～100%），第二个参数值表示任意颜色值。

【示例 2】下面示例针对示例 1，兼容早期 Webkit 引擎的线性渐变实现方法。

```
#demo {
    width:300px; height:200px;
    background: -webkit-gradient(linear, left top, left bottom, from(#fff), to(#333));
    background: linear-gradient(#fff, #333);
}
```

上面示例定义线性渐变背景色，从顶部到底部，从白色向浅灰色渐变显示，在谷歌的 Chrome 浏览器中所见效果与图 5.11 相同。

另外，Webkit 引擎也支持-webkit-linear-gradient()私有函数来设计线性渐变。该函数用法与标准函数 linear-gradient()语法格式基本相同。

Firefox 浏览器从 3.6 版本开始支持渐变，Gecko 引擎定义了-moz-linear-gradient()私有函数来设计线性渐变。该函数用法与标准函数 linear-gradient()语法格式基本相同。唯一区别是，当使用关键字设置渐变方向时，不带 to 关键字前缀，关键字语义取反。例如，从上到下应用渐变，标准关键字为 to bottom，Firefox 私有属性可以为 top。

【示例 3】下面示例针对示例 1，兼容早期 Gecko 引擎的线性渐变实现方法。



```
#demo {
    width:300px; height:200px;
    background: -webkit-gradient(linear, left top, left bottom, from(#fff), to(#333));
    background: -moz-linear-gradient(top, #fff, #333);
    background: linear-gradient(#fff, #333);
}
```



Note



视频讲解

5.2.2　设计线性渐变样式

本节以示例形式介绍线性渐变中渐变方向和色点的设置，演示设计线性渐变的一般方法。

【示例 1】下面示例演示了从左边开始的线性渐变。起点是红色，慢慢过渡到蓝色（可扫描本书“视频讲解”二维码查看颜色效果，后面不再一一标注），效果如图 5.12 所示。

```
<style type="text/css">
#demo {
    width:300px; height:200px;
    background: -webkit-linear-gradient(left, red , blue);      /*Safari 5.1 - 6.0*/
    background: -o-linear-gradient(left, red, blue);            /*Opera 11.1 - 12.0*/
    background: -moz-linear-gradient(left, red, blue);          /*Firefox 3.6 - 15*/
    background: linear-gradient(to right, red , blue);          /*标准语法*/
}
</style>
<div id="demo"></div>
```

注意：第一个参数值渐变方向的设置不同。

【示例 2】通过指定水平和垂直的起始位置来设计对角渐变。下面示例演示了从左上角到右下角的线性渐变，起点是红色，慢慢过渡到蓝色，效果如图 5.13 所示。

```
#demo {
    width:300px; height:200px;
    background: -webkit-linear-gradient(left top, red , blue);   /*Safari 5.1 - 6.0*/
    background: -o-linear-gradient(left top, red, blue);         /*Opera 11.1 - 12.0*/
    background: -moz-linear-gradient(left top, red, blue);       /*Firefox 3.6 - 15*/
    background: linear-gradient(to bottom right, red , blue);    /*标准语法*/
}
```

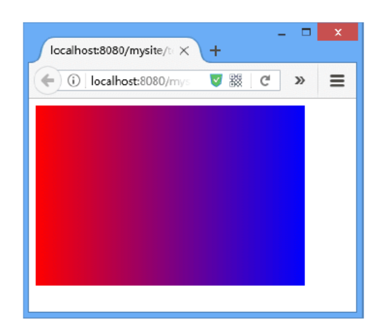


图 5.12　设计从左到右的线性渐变效果

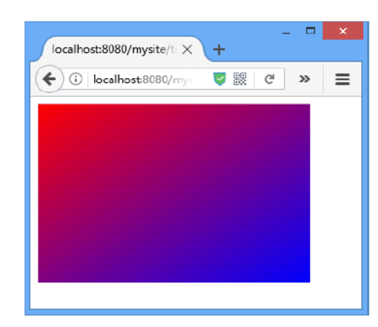


图 5.13　设计对角线性渐变效果

【示例 3】通过指定具体的角度值，可以设计更多渐变方向。下面示例演示了从上到下的线性渐变，起点是红色，慢慢过渡到蓝色，效果如图 5.14 所示。





Note

```
#demo {
    width:300px; height:200px;
    background: -webkit-linear-gradient(-90deg, red, blue);        /*Safari 5.1 - 6.0*/
    background: -o-linear-gradient(-90deg, red, blue);             /*Opera 11.1 - 12.0*/
    background: -moz-linear-gradient(-90deg, red, blue);           /*Firefox 3.6 - 15*/
    background: linear-gradient(180deg, red, blue);                /*标准语法*/
}
```

【补充】

渐变角度是指垂直线和渐变线之间的角度，逆时针方向计算。例如，0deg 将创建一个从下到上的渐变，90deg 将创建一个从左到右的渐变。注意，渐变起点以-y 轴为参考。

但是，很多浏览器（如 Chrome、Safari、Firefox 等）使用旧的标准：渐变角度是指水平线和渐变线之间的角度，逆时针方向计算。例如，0deg 将创建一个从左到右的渐变，90deg 将创建一个从下到上的渐变。注意，渐变起点以-x 轴为参考。

兼容公式：

```
90 - x = y
```

其中，x 为标准角度，y 为非标准角度。

【示例 4】设置多个色点。下面示例定义从上到下的线性渐变，起点是红色，慢慢过渡到绿色，再慢慢过渡到蓝色，效果如图 5.15 所示。

```
#demo {
    width:300px; height:200px;
    background: -webkit-linear-gradient(red, green, blue);         /*Safari 5.1 - 6.0*/
    background: -o-linear-gradient(red, green, blue);              /*Opera 11.1 - 12.0*/
    background: -moz-linear-gradient(red, green, blue);            /*Firefox 3.6 - 15*/
    background: linear-gradient(red, green, blue);                 /*标准语法*/
}
```

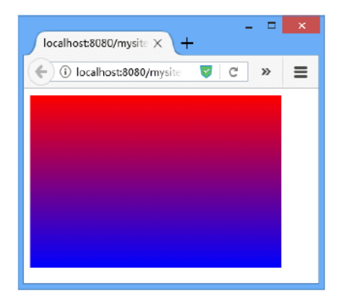


图 5.14　设计从上到下的渐变效果

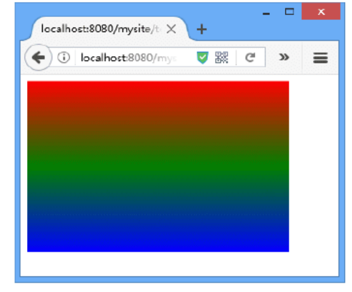


图 5.15　设计多色线性渐变效果

【示例 5】设置色点位置。下面示例定义从上到下的线性渐变，起点是黄色，快速过渡到蓝色，再慢慢过渡到绿色，效果如图 5.16 所示。

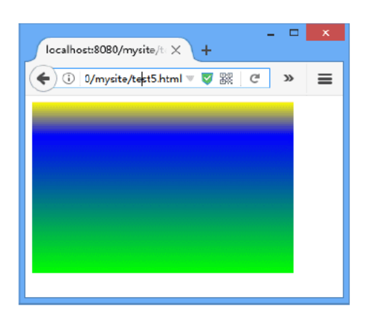
```
#demo {
    width:300px; height:200px;
    background: -webkit-linear-gradient(yellow, blue 20%, #0f0);   /*Safari 5.1 - 6.0*/
    background: -o-linear-gradient(yellow, blue 20%, #0f0);        /*Opera 11.1 - 12.0*/
    background: -moz-linear-gradient(yellow, blue 20%, #0f0);      /*Firefox 3.6 - 15*/
    background: linear-gradient(yellow, blue 20%, #0f0);           /*标准语法*/
}
```



Note

【示例 6】CSS3 渐变支持透明度设置，可用于创建减弱变淡的效果。下面示例演示了从左边开始的线性渐变。起点是完全透明，起点位置为 30%，慢慢过渡到完全不透明的红色，为了更清晰地看到半透明效果，示例增加了一层背景图像进行衬托，演示效果如图 5.17 所示。

```
#demo {
    width:300px; height:200px;
    /*Safari 5.1 - 6*/
    background: -webkit-linear-gradient(left,rgba(255,0,0,0) 30%,rgba(255,0,0,1)),url(images/bg.jpg);
    /*Opera 11.1 - 12*/
    background: -o-linear-gradient(left,rgba(255,0,0,0) 30%,rgba(255,0,0,1)),url(images/bg.jpg);
    /*Firefox 3.6 - 15*/
    background: -moz-linear-gradient(left,rgba(255,0,0,0) 30%,rgba(255,0,0,1)),url(images/bg.jpg);
    /*标准语法*/
    background: linear-gradient(to right, rgba(255,0,0,0) 30%, rgba(255,0,0,1)),url(images/bg.jpg);
    background-size:cover;              /*背景图像完全覆盖*/
}
```



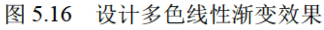


图 5.16　设计多色线性渐变效果

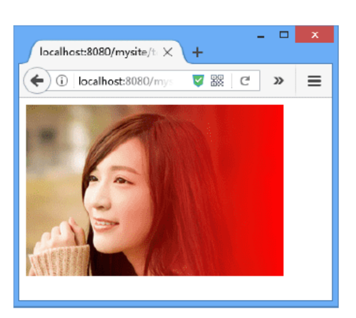


图 5.17　设计半透明线性渐变效果

提示：为了添加透明度，可以使用 rgba() 或 hsla()函数来定义色点。rgba()或 hsla()函数中最后一个参数可以是 0～1 的值，它定义了颜色的透明度：0 表示完全透明，1 表示完全不透明。

5.2.3　案例：设计网页渐变色



视 频 讲 解

为页面设计渐变背景，可以营造特殊的浏览气氛。本例主要代码如下所示，预览效果如图 5.18 所示。

```
<style type="text/css">
body { /*让渐变背景填满整个页面*/
    padding: 1em;
    margin: 0;
    background: -webkit-linear-gradient(#FF6666, #ffffff);          /*Safari 5.1 - 6.0*/
    background: -o-linear-gradient(#FF6666, #ffffff);               /*Opera 11.1 - 12.0*/
    background: -moz-linear-gradient(#FF6666, #ffffff);             /*Firefox 3.6 - 15*/
    background: linear-gradient(#FF6666, #ffffff);                  /*标准语法*/
    /*IE 滤镜，兼容 IE9-版本浏览器*/
 filter: progid:DXImageTransform.Microsoft.Gradient(gradientType=0, startColorStr=#FF6666, endColorStr=#ffffff);
}
h1 {/*定义标题样式*/
```





```
        color: white;
        font-size: 18px;
        height: 45px;
        padding-left: 3em;
        line-height: 50px;                               /*控制文本显示位置*/
        border-bottom: solid 2px red;
        background: url(images/pe1.png) no-repeat left center;        /*为标题插入一个装饰图标*/
    }
    p { text-indent: 2em; }/*段落文本缩进 2 个字符*/
    </style>
    <div  class="box">
        <h1>W3C 发布 HTML5 的正式推荐标准</h1>
        <p>2014 年 10 月 28 日，W3C 的 HTML 工作组正式发布了 HTML5 的正式推荐标准（W3C Recommendation）。W3C 在美国圣克拉拉举行的 W3C 技术大会及顾问委员会会议（TPAC 2014）上宣布了这一消息。 HTML5 是万维网的核心语言——可扩展标记语言的第 5 版。在这一版本中，增加了支持 Web 应用开发者的许多新特性，以及更符合开发者使用习惯的新元素，并重点关注定义清晰的、一致的准则，以确保 Web 应用和内容在不同用户代理（浏览器）中的互操作性。HTML5 是构建开放 Web 平台的核心。</p>
        <p class="right">更多<a href="http://www.chinaw3c.org/archives/677/" target="_blank">详细内容</a></p>
    </div>
```

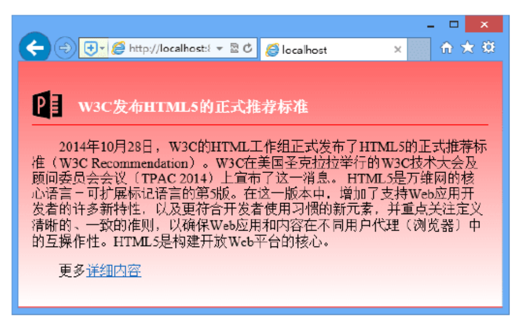


图 5.18　设计渐变网页背景色效果

【补充】

IE 早期版本不支持 CSS 渐变，但提供了渐变滤镜，可以实现简单的渐变效果。IE 浏览器渐变滤镜的基本语法说明如下：

```
filter:progid:DXImageTransform.Microsoft.Gradient(enabled=bEnabled,startColorStr=iWidth,endColorStr=iWidth)
```

该函数的参数说明如下。

☑ enabled：设置或检索滤镜是否激活。可选布尔值，包括 true 和 false，默认值为 true，即激活状态。

☑ startColorStr：设置或检索色彩渐变的开始颜色和透明度。可选项，其格式为#AARRGGBB。AA、RR、GG、BB 为十六进制正整数，取值范围为 00～FF。RR 指定红色值，GG 指定绿色值，BB 指定蓝色值。AA 指定透明度，00 是完全透明，FF 是完全不透明。超出取值范围的值将被恢复为默认值。取值范围为#FF000000～#FFFFFFFF，默认值为#FF0000FF，即不透明蓝色。

☑ endColorStr：设置或检索色彩渐变的结束颜色和透明度。默认值为#FF000000，即不透明黑色。

注意：IE 渐变滤镜在 IE 5.5 及其以上版本浏览器中有效。



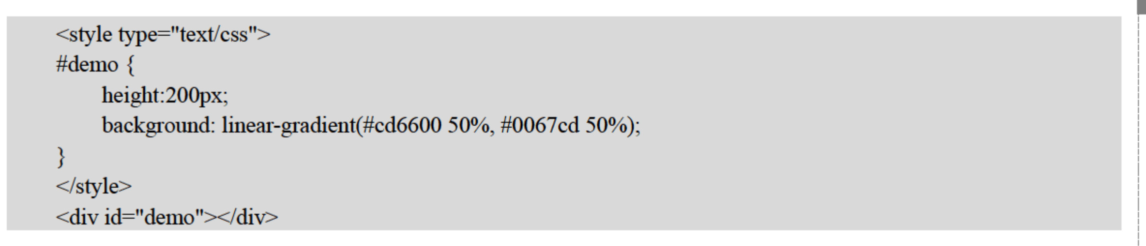




5.2.4　案例：设计条纹背景

如果多个色点设置为相同的起点位置，它们将产生从一种颜色到另一种颜色的急剧转换。从效果来看，就是从一种颜色突然改变到另一种颜色，这样可以设计条纹背景效果。

【示例 1】定义一个简单的条纹背景，效果如图 5.19 所示。



```
<style type="text/css">
#demo {
    height:200px;
    background: linear-gradient(#cd6600 50%, #0067cd 50%);
}
</style>
<div id="demo"></div>
```

【示例 2】利用背景的重复机制，可以创造出更多的条纹。示例代码如下所示，效果如图 5.20 所示。这样就可以将整个背景划分为 10 个条纹，每个条纹的高度一样。

```
#demo {
    height:200px;
    background: linear-gradient(#cd6600 50%, #0067cd 50%);
    background-size: 100% 20%;                    /*定义单个条纹仅显示高度的五分之一*/
}
```

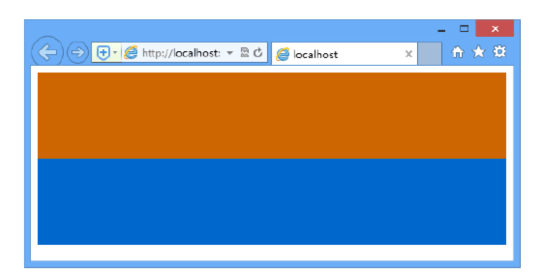


图 5.19　设计简单的条纹效果

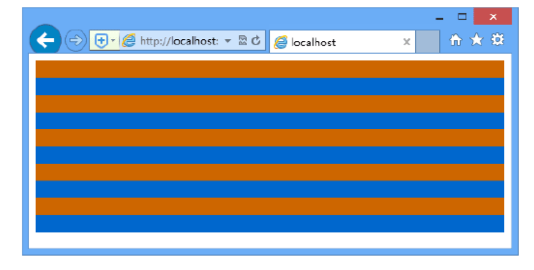


图 5.20　设计重复显示的条纹效果

【示例 3】如果设计每个条纹高度不同，只要改变比例即可。示例代码如下所示，效果如图 5.21 所示。

```
#demo {
    height:200px;
    background: linear-gradient(#cd6600 80%, #0067cd 0%);  /*定义每个条纹位置占比不同*/
    background-size: 100% 20%;                    /*定义单个条纹仅显示高度的五分之一*/
}
```

【示例 4】设计多色条纹背景，代码如下所示，效果如图 5.22 所示。

```
#demo {
    height:200px;
    /*定义三色同宽背景*/
    background: linear-gradient(#cd6600 33.3%, #0067cd 0, #0067cd 66.6%, #00cd66 0);
    background-size: 100% 30px;
}
```




Note

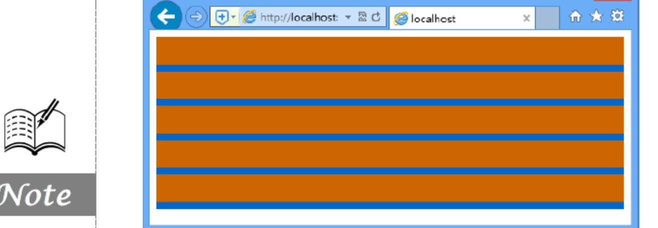


图 5.21　设计不同高度的条纹效果

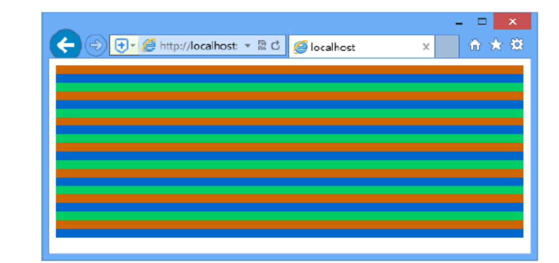


图 5.22　设计多色条纹效果

【示例 5】设计密集条纹格效果，代码如下所示，效果如图 5.23 所示。

```
#demo {
    height:200px;
    background: linear-gradient(rgba(0,0,0,.5) 1px, #fff 1px);
    background-size: 100% 3px;
}
```

注意：IE 不支持这种设计效果。

【示例 6】设计垂直条纹背景，只需要转换一下宽和高的设置方式，具体代码如下所示，效果如图 5.24 所示。

```
#demo {
    height:200px;
    background: linear-gradient(to right, #cd6600 50%, #0067cd 0);
    background-size: 20% 100%;
}
```

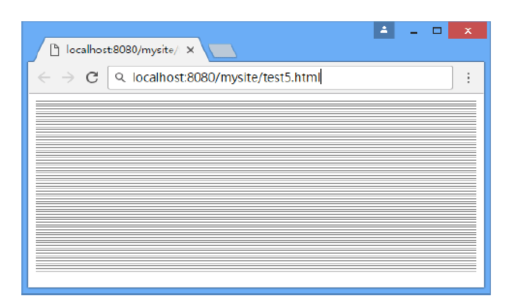


图 5.23　设计密集条纹效果

图 5.24　设计垂直条纹效果

【示例 7】设计简单的纹理背景，代码如下所示，效果如图 5.25 所示。

```
#demo {
    height:200px;
    background: linear-gradient(45deg, RGBA(0,103,205,0.2)   50%, RGBA(0,103,205,0.1)   50%);
    background-size: 50px 50px;
}
```

提示：在实际应用中，不建议使用太多的背景颜色，一般可以考虑使用一种背景色，并在这个颜色的深浅上设计变化。



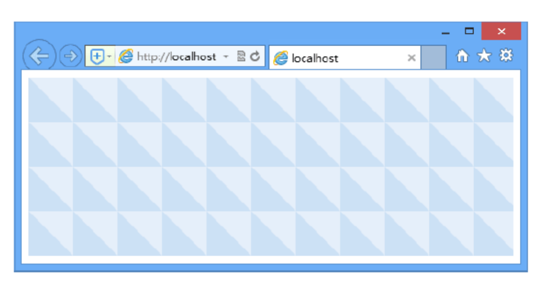


图 5.25　设计简单的纹理效果





视 频 讲 解

5.2.5　定义重复线性渐变

使用 repeating-linear-gradient()函数可以定义重复线性渐变，用法与 linear-gradient()函数相同，用户可以参考 5.2.1 节说明。

提示： 使用重复线性渐变的关键是要定义好色点，让最后一个颜色和第一个颜色能够很好地连接起来，处理不当将导致颜色的急剧变化。

【示例 1】 下面示例设计重复显示的垂直线性渐变，颜色从红色到蓝色，间距为 20%，效果如图 5.26 所示。

```
<style type="text/css">
#demo {
    height:200px;
    background: repeating-linear-gradient(#f00, #00f 20%, #f00 40%);
}
</style>
<div id="demo"></div>
```

提示： 使用 linear-gradient()可以设计 repeating-linear-gradient()的效果。例如，通过重复设计每一个色点或者利用 5.2.4 节设计条纹的方法来实现。

【示例 2】 下面示例设计重复线性渐变对角显示，效果如图 5.27 所示。

```
#demo {
    height:200px;
    background: repeating-linear-gradient(135deg, #cd6600, #0067cd 20px, #cd6600 40px);
}
```



图 5.26　设计重复显示的垂直渐变效果

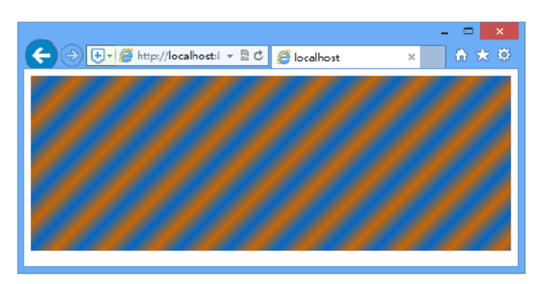


图 5.27　设计重复显示的对角渐变效果

【示例 3】 下面示例使用重复线性渐变创建出对角条纹背景，效果如图 5.28 所示。



```
#demo {
    height:200px;
    background: repeating-linear-gradient(60deg, #cd6600, #cd6600 5%, #0067cd 0, #0067cd 10%);
}
```





图 5.28　设计重复显示的对角条纹效果



视频讲解

5.2.6　定义径向渐变

创建一个径向渐变，也至少需要定义两个颜色，同时可以指定渐变的中心点位置、形状类型（圆形或椭圆形）和半径大小。简明语法格式如下：

```
radial-gradient(shape size at position, color-stop1, color-stop2, …);
```

参数简单说明如下。

☑ shape：用来指定渐变的类型，包括 circle（圆形）和 ellipse（椭圆）两种。

☑ size：如果类型为 circle，指定一个值设置圆的半径；如果类型为 ellipse，指定两个值分别设置椭圆的 x 轴和 y 轴半径。取值包括长度值、百分比、关键字。关键字说明如下。

- ➢ closest-side：指定径向渐变的半径长度为从中心点到最近的边。
- ➢ closest-corner：指定径向渐变的半径长度为从中心点到最近的角。
- ➢ farthest-side：指定径向渐变的半径长度为从中心点到最远的边。
- ➢ farthest-corner：指定径向渐变的半径长度为从中心点到最远的角。

☑ position：用来指定中心点的位置。如果提供两个参数，第一个表示 x 轴坐标，第二个表示 y 轴坐标；如果只提供一个值，第二个值默认为 50%，即 center。取值可以是长度值、百分比或者关键字，关键字包括 left（左侧）、center（中心）、right（右侧）、top（顶部）、center（中心）、bottom（底部）。

注意：position 值位于 shape 和 size 值后面。

☑ color-stop：用于指定渐变的色点。包括一个颜色值和一个起点位置，颜色值和起点位置以空格分隔。起点位置可以为一个具体的长度值（不可为负值），也可以是一个百分比值，如果是百分比值则参考应用渐变对象的尺寸，最终会被转换为具体的长度值。

【示例 1】在默认情况下，渐变的中心是 center（对象中心点），渐变的形状是 ellipse（椭圆形），渐变的大小是 farthest-corner（表示到最远的角落）。下面示例仅为 radial-gradient()函数设置 3 个颜色值，则它将按默认值绘制径向渐变效果，如图 5.29 所示。

```
<style type="text/css">
#demo {
    height:200px;
    background: -webkit-radial-gradient(red, green, blue);        /*Safari 5.1 - 6.0*/
```



```
        background: -o-radial-gradient(red, green, blue);          /*Opera 11.6 - 12.0*/
        background: -moz-radial-gradient(red, green, blue);        /*Firefox 3.6 - 15*/
        background: radial-gradient(red, green, blue);             /*标准语法*/
    }
    </style>
    <div id="demo"></div>
```



Note

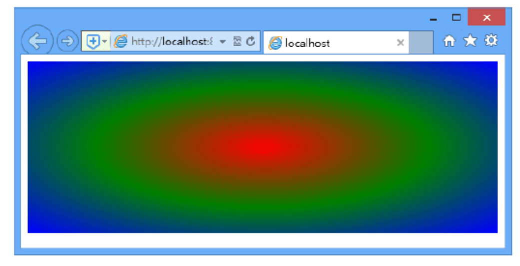


图 5.29　设计简单的径向渐变效果

提示：针对示例 1，用户可以继续尝试做下面练习，实现不同的设置，得到相同的设计效果。

☑　设置径向渐变形状类型，默认值为 ellipse。

```
background: radial-gradient(ellipse, red, green, blue);
```

☑　设置径向渐变中心点坐标，默认为对象中心点。

```
background: radial-gradient(ellipse at center 50%, red, green, blue);
```

☑　设置径向渐变大小，这里定义填充整个对象。

```
background: radial-gradient(farthest-corner, red, green, blue);
```

【拓展】

最新主流浏览器都支持径向渐变的标准用法，但是考虑到安全性，用户应酌情兼容旧版本浏览器的私有属性。

Webkit 引擎使用-webkit-gradient()私有函数支持径向渐变样式，简明用法如下：

```
-webkit-gradient(radial, point, radius, stop)
```

参数简单说明如下。

☑　radial：定义渐变类型为径向渐变。

☑　point：定义渐变中心点坐标。该参数支持数值、百分比值和关键字，如(0 0)或者(left top)等。关键字包括 top、bottom、center、left 和 right。

☑　radius：设置径向渐变的长度，该参数为一个数值。

☑　stop：定义渐变色和步长。包括 3 个值，即开始的颜色，使用 from(colorvalue)函数定义；结束的颜色，使用 to(colorvalue)函数定义；颜色步长，使用 color-stop(value, color value)函数定义。color-stop()函数包含两个参数值，第一个参数值为一个数值或者百分比值，取值范围为 0～1.0（或者 0%～100%），第二个参数值表示任意颜色值。

【示例 2】下面示例设计一个红色圆球，并逐步径向渐变为绿色背景，兼容早期 Webkit 引擎的线性渐变实现方法。代码如下所示，演示效果如图 5.30 所示。

```
<style type="text/css">
#demo {
```



```
        height:200px;
        /*Webkit 引擎私有用法*/
        background: -webkit-gradient(radial, center center, 0, center center, 100, from(red), to(green));
        background: radial-gradient(circle 100px, red, green);        /*标准的用法*/
}
</style>
<div id="demo"></div>
```

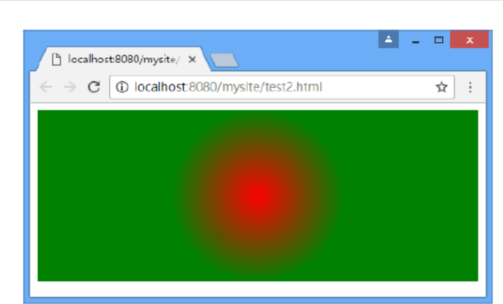


图 5.30　设计径向圆球效果

另外，Webkit 引擎也支持-webkit-radial-gradient()私有函数来设计径向渐变。该函数用法与标准函数 radial-gradient()语法格式类似。简明语法格式如下：

```
-webkit-radial-gradient(position, shape size, color-stop1, color-stop2, …);
```

Gecko 引擎定义了-moz-radial-gradient()私有函数来设计径向渐变。该函数用法与标准函数 radial-gradient()语法格式也类似。简明语法格式如下：

```
-moz-radial-gradient(position, shape size, color-stop1, color-stop2, …);
```

提示：上面两个私有函数的 size 参数值仅可设置为关键字 closest-side、closest-corner、farthest-side、farthest-corner、contain 或 cover。



视频讲解

5.2.7　设计径向渐变样式

本节以示例形式介绍径向渐变的灵活设置，演示设计径向渐变的一般方法。

【示例 1】下面示例演示了色点不均匀分布的径向渐变，效果如图 5.31 所示。

```
<style type="text/css">
#demo {
        height:200px;
        background: -webkit-radial-gradient(red 5%, green 15%, blue 60%);    /*Safari 5.1 - 6.0*/
        background: -o-radial-gradient(red 5%, green 15%, blue 60%);         /*Opera 11.6 - 12.0*/
        background: -moz-radial-gradient(red 5%, green 15%, blue 60%);       /*Firefox 3.6 - 15*/
        background: radial-gradient(red 5%, green 15%, blue 60%);            /*标准语法*/
}
</style>
<div id="demo"></div>
```

【示例 2】shape 参数定义了形状，取值包括 circle 和 ellipse，其中 circle 表示圆形，ellipse 表示



椭圆形，默认值是 ellipse。下面示例设计圆形径向渐变，效果如图 5.32 所示。



```
#demo {
    height:200px;
    background: -webkit-radial-gradient(circle, red, yellow, green);   /*Safari 5.1 - 6.0*/
    background: -o-radial-gradient(circle, red, yellow, green);        /*Opera 11.6 - 12.0*/
    background: -moz-radial-gradient(circle, red, yellow, green);      /*Firefox 3.6 - 15*/
    background: radial-gradient(circle, red, yellow, green);           /*标准语法*/
}
```

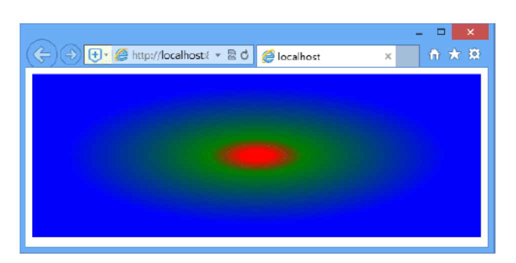


图 5.31　设计色点不均匀分布的径向渐变效果

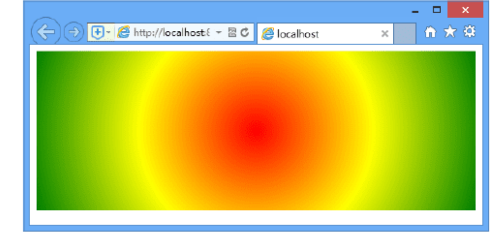


图 5.32　设计圆形径向渐变效果

【示例 3】下面设计径向渐变的半径长度为从圆心到离圆心最近的边，效果如图 5.33 所示。

```
#demo {
    height:200px;
    /*Safari 5.1 - 6.0*/
    background: -webkit-radial-gradient(60% 55%, closest-side,blue,green,yellow,black);
    /*Opera 11.6 - 12.0*/
    background: -o-radial-gradient(60% 55%, closest-side,blue,green,yellow,black);
    /*Firefox 3.6 - 15*/
    background: -moz-radial-gradient(60% 55%, closest-side,blue,green,yellow,black);
    /*标准语法*/
    background: radial-gradient(closest-side at 60% 55%, blue,green,yellow,black);
}
```

注意：radial-gradient()标准函数与各私有函数在设置参数时的顺序区别。

【示例 4】下面示例模拟太阳初升的效果，如图 5.34 所示。设计径向渐变中心点位于左下角，半径为最大化显示。定义 3 个色点，第 1 个色点设计太阳效果，第 2 个色点设计太阳余晖，第 3 个色点设计太空，第 1 个色点和第 2 个色点距离为 60px。

```
#demo {
    height:200px;
    /*Safari 5.1 - 6.0*/
    background: -webkit-radial-gradient(left bottom, farthest-side, #f00, #f99 60px, #005);
    /*Opera 11.6 - 12.0*/
    background: -o-radial-gradient(left bottom, farthest-side, #f00, #f99 60px, #005);
    /*Firefox 3.6 - 15*/
    background: -moz-radial-gradient(left bottom, farthest-side, #f00, #f99 60px, #005);
    /*标准语法*/
    background: radial-gradient(farthest-side at left bottom, #f00, #f99 60px, #005);
}
```



Note



图 5.33　设计最小限度的径向渐变效果



图 5.34　模拟太阳初升效果

【示例 5】下面示例模拟使用径向渐变绘制圆形图效果，如图 5.35 所示。设计径向渐变中心点位于对象中央，定义两个色点，第 1 个色点设计圆形效果，第 2 个色点设计背景，两个色点位置相同。

```
<style type="text/css">
body {background:hsla(207,59%,78%,1.00)}
#demo {
    height:200px;
    width:300px;
    margin:auto;
    /*Safari 5.1 - 6.0*/
    background: -webkit-radial-gradient(center, circle, #f00 50px, #fff 50px);
    /*Opera 11.6 - 12.0*/
    background: -o-radial-gradient(center, circle, #f00 50px, #fff 50px);
    /*Firefox 3.6 - 15*/
    background: -moz-radial-gradient(center, circle, #f00 50px, #fff 50px);
    /*标准语法*/
    background: radial-gradient(circle  at center, #f00 50px, #fff 50px);
}
</style>
<div id="demo"></div>
```

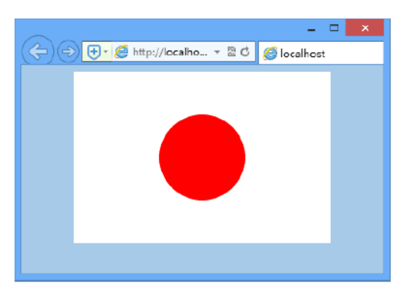


图 5.35　设计圆形效果



视频讲解

5.2.8　定义重复径向渐变

使用 repeating-radial-gradient()函数可以定义重复线性渐变，用法与 radial-gradient()函数相同，用户可以参考前面说明。

【示例 1】下面示例设计三色重复显示的径向渐变，效果如图 5.36 所示。

```
<style type="text/css">
#demo {
```



```
        height:200px;
        /*Safari 5.1 - 6.0*/
        background: -webkit-repeating-radial-gradient(red, yellow 10%, green 15%);
        /*Opera 11.6 - 12.0*/
        background: -o-repeating-radial-gradient(red, yellow 10%, green 15%);
        /*Firefox 3.6 - 15*/
        background: -moz-repeating-radial-gradient(red, yellow 10%, green 15%);
        /*标准语法*/
        background: repeating-radial-gradient(red, yellow 10%, green 15%);
    }
    </style>
    <div id="demo"></div>
```



【示例 2】使用径向渐变同样可以创建条纹背景，方法与线性渐变类似。下面示例设计圆形径向渐变条纹背景，效果如图 5.37 所示。

```
    #demo {
        height:200px;
        /*Safari 5.1 - 6.0*/
        background: -webkit-repeating-radial-gradient(center bottom, circle, #00a340, #00a340 20px, #d8ffe7 20px, 
#d8ffe7 40px);
        /*Opera 11.6 - 12.0*/
        background: -o-repeating-radial-gradient(center bottom, circle, #00a340, #00a340 20px, #d8ffe7 20px, 
#d8ffe7 40px);
        /*Firefox 3.6 - 15*/
        background: -moz-repeating-radial-gradient(center bottom, circle, #00a340, #00a340 20px, #d8ffe7 20px, 
#d8ffe7 40px);
        /*标准语法*/
        background: repeating-radial-gradient(circle at center bottom, #00a340, #00a340 20px, #d8ffe7 20px, #d8ffe7 40px);
    }
```

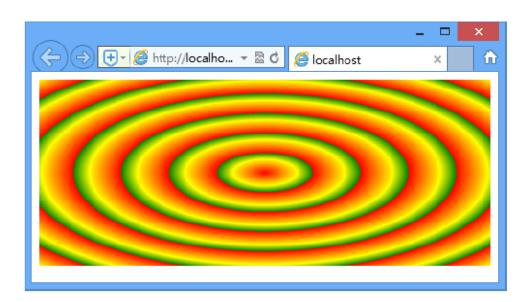


图 5.36　设计重复显示的径向渐变效果

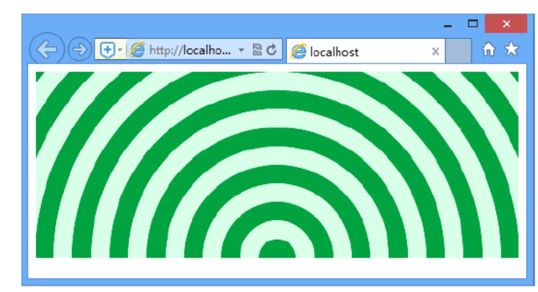


图 5.37　设计径向渐变条纹背景效果

5.2.9　案例：设计网页背景色



视频讲解

【示例 1】为页面叠加多个径向渐变背景，可以营造虚幻的页面氛围。示例代码如下所示，预览效果如图 5.38 所示。

```
    <style type="text/css">
    html, body{height:100%;}
    body {
        background-color: #4B770A;
```



Note

```
        background-image:
            radial-gradient( rgba(255, 255, 255, 0.3), rgba(255, 255, 255, 0)),
            radial-gradient(at 10% 5%, rgba(255, 255, 255, 0.1), rgba(255, 255, 255, 0) 20%),
            radial-gradient(at left bottom , rgba(255, 255, 255, 0.2), rgba(255, 255, 255, 0) 20%),
            radial-gradient(at right top, rgba(255, 255, 255, 0.2), rgba(255, 255, 255, 0) 20%),
            radial-gradient(at 85% 90% , rgba(255, 255, 255, 0.1), rgba(255, 255, 255, 0) 20%);
}
</style>
```

在上面示例代码中，首先设计 body 高度满屏显示，避免无内容时看不到效果；然后为页面定义一个基本色#4B770A；再设计 5 个径向渐变，分别散布于页面 4 个顶角和中央位置，同时定义径向渐变的第 1 个颜色为半透明的白色，第 2 个颜色为透明色，从而在页面不同位置蒙上轻重不一的白粉效果，以此来模拟虚幻莫测的背景效果。

【示例 2】为页面叠加 4 个径向渐变背景，设计密密麻麻的针脚纹理效果。示例代码如下所示，预览效果如图 5.39 所示。

```
<style type="text/css">
html, body{height:100%;}
body {
    background-color: #282828;
    background-image:
        -webkit-radial-gradient(black 15%, transparent 16%),
        -webkit-radial-gradient(black 15%, transparent 16%),
        -webkit-radial-gradient(rgba(255, 255, 255, 0.1) 15%, transparent 20%),
        -webkit-radial-gradient(rgba(255, 255, 255, 0.1) 15%, transparent 20%);
    background-image:
        radial-gradient(black 15%, transparent 16%),
        radial-gradient(black 15%, transparent 16%),
        radial-gradient(rgba(255, 255, 255, 0.1) 15%, transparent 20%),
        radial-gradient(rgba(255, 255, 255, 0.1) 15%, transparent 20%);
    background-position:
        0 0px,
        8px 8px,
        0 1px,
        8px 9px;
    background-size: 16px 16px;
}
</style>
```



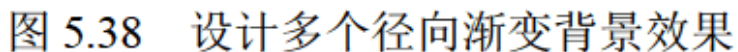


图 5.38　设计多个径向渐变背景效果

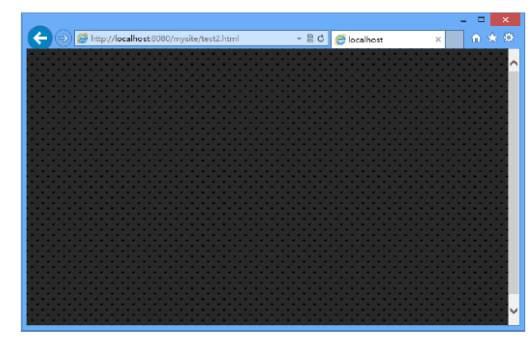


图 5.39　设计针脚纹理背景效果



在上面示例代码中，首先使用 background-size: 16px 16px;定义背景图大小为 16px×16px；然后在这块小图上设计 4 个径向渐变，包括两个深色径向渐变和两个浅色径向渐变；再使用 background-position: 0 0px, 8px 8px, 0 1px, 8px 9px;设计一深、一浅径向渐变错位叠加，在 y 轴上错位 1px，从而在 16×16 大小的浅色背景图上设计了两个深色凹陷效果；最后，借助背景图平铺，为网页设计上述纹理特效。



视频讲解

5.2.10　案例：设计按钮

【示例 1】 利用 CSS3 线性渐变设计立体的按钮效果，同时定义当按钮激活或者鼠标经过时，调整线性渐变的方向，实现动态提示响应。示例代码如下所示，演示效果如图 5.40 所示。

```
<style type="text/css">
/*按钮默认类样式*/
.button {
    /*为按钮底部添加浅色的阴影*/
    -moz-box-shadow:inset 0px 1px 0px 0px #ffffff;
    -webkit-box-shadow:inset 0px 1px 0px 0px #ffffff;
    box-shadow:inset 0px 1px 0px 0px #ffffff;
    /*设计从上到下线性渐变，颜色为半透明的浅灰色到半透明的浅白色*/
    background:-webkit-gradient(linear, left top, left bottom, color-stop(0.05, #ededed), color-stop(1, #dfdfdf));
    /*谷歌传统用法*/
    background:-moz-linear-gradient(top, #ededed 5%, #dfdfdf 100%);
    background:-webkit-linear-gradient(top, #ededed 5%, #dfdfdf 100%);  /*谷歌标准用法*/
    background:-o-linear-gradient(top, #ededed 5%, #dfdfdf 100%);
    background:-ms-linear-gradient(top, #ededed 5%, #dfdfdf 100%);      /*兼容 IE9*/
    background:linear-gradient(to bottom, #ededed 5%, #dfdfdf 100%);
    filter:progid:DXImageTransform.Microsoft.gradient(startColorstr='#ededed', endColorstr='#dfdfdf',GradientType=0);
                                                                       /*IE 滤镜特效*/
    /*定义背景基色*/
    background-color:#ededed;
    /*设计圆角效果*/
    -moz-border-radius:6px;
    -webkit-border-radius:6px;
    border-radius:6px;
    /*定义浅色边框线*/
    border:1px solid #dcdcdc;
    display:inline-block;                                              /*行内块显示*/
    cursor:pointer;                                                    /*鼠标经过显示手形指针样式*/
    color:#777777;                                                     /*字体中灰色显示*/
    font-family:arial;
    font-size:16px;
    font-weight:bold;
    padding:12px 24px;                                                 /*调整按钮大小*/
    text-decoration:none;                                              /*清除可能存在的下画线样式*/
    text-shadow:0px 1px 0px #ffffff;                                   /*为按钮文本添加浅色阴影*/
}
/*鼠标经过时按钮类样式*/
.button:hover {
    background:-webkit-gradient(linear, left top, left bottom, color-stop(0.05, #dfdfdf), color-stop(1, #ededed));
    background:-moz-linear-gradient(top, #dfdfdf 5%, #ededed 100%);
    background:-webkit-linear-gradient(top, #dfdfdf 5%, #ededed 100%);
    background:-o-linear-gradient(top, #dfdfdf 5%, #ededed 100%);
```



Note

```
        background:-ms-linear-gradient(top, #dfdfdf 5%, #ededed 100%);
        background:linear-gradient(to bottom, #dfdfdf 5%, #ededed 100%);
        filter:progid:DXImageTransform.Microsoft.gradient(startColorstr='#dfdfdf', endColorstr='#ededed',GradientType=0);
        background-color:#dfdfdf;
}
/*被激活时按钮类样式*/
.button:active {
        position:relative;                    /*相对定位，方便移位*/
        top:1px;                              /*下移 1px*/
}
</style>
<a href="#"   class="button">使用线性渐变设计立体动态按钮效果</a>
```



默认从上到下渐变



鼠标经过时从下到上渐变



激活时下移 1px

图 5.40　设计按钮效果

【示例 2】本示例设计按钮在正常状态下带有边框且有渐变和阴影效果，当鼠标经过时会显示比较暗的渐变背景效果，当按下鼠标时会翻转渐变背景，并显示 1px 的下沉效果，按钮字体颜色加深。演示效果如图 5.41 所示。

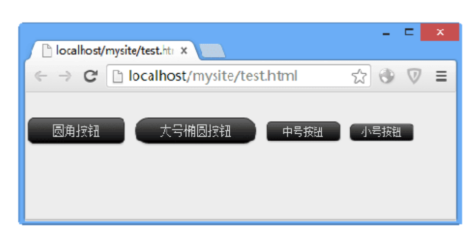


图 5.41　设计精致的按钮

示例代码如下：

```
<style type="text/css">
/*设计页面基本样式：浅色背景、网页居中、浅灰字体*/
body {background:#ededed; margin: 30px auto; color: #999;}
.button {                                     /*定义渐变按钮样式类*/
        display: inline-block;
        /*zoom 和 *display 属性都为了兼容 IE7，使其具有 display:inlineblock 特性*/
        zoom: 1;
        *display: inline;
        vertical-align: baseline;             /*垂直基线对齐*/
        margin: 0 2px;                        /*左右加 2px 边界，避免按钮挤在一起*/
        outline: none;                        /*清除轮廓线，针对 a 和表单按钮*/
        cursor: pointer;                      /*增加鼠标经过时显示手形指针*/
        text-align: center;                   /*文本水平对齐*/
```



```
        text-decoration: none;                  /*清除下画线，针对 a*/
        font: 14px/100% Arial, Helvetica, sans-serif;
        padding: .5em 2em .55em;                /*调整按钮内补白，让按钮看起来更大方*/
        /*设计按钮圆角、盒子阴影和文本阴影特效*/
        text-shadow: 0 1px 1px rgba(0, 0, 0, .3);
        -webkit-border-radius: .5em;
        -moz-border-radius: .5em;
        border-radius: .5em;                    /*精致圆角*/
        -webkit-box-shadow: 0 1px 2px rgba(0, 0, 0, .2);
        -moz-box-shadow: 0 1px 2px rgba(0, 0, 0, .2);
        box-shadow: 0 1px 2px rgba(0, 0, 0, .2);  /*添加淡淡阴影*/
    }
    .button:hover {text-decoration: none;}      /*鼠标经过清除下画线，针对 a*/
    .button:active {                            /*按钮被激活时，下沉 1px，模拟凹陷效果*/
        position: relative;
        top: 1px;
    }
    .bigrounded {                               /*定义大圆角样式类*/
        -webkit-border-radius: 2em;
        -moz-border-radius: 2em;
        border-radius: 2em;
    }
    .medium {                                   /*定义大按钮样式类*/
        font-size: 12px;
        padding: .4em 1.5em .42em;
    }
    .small {                                    /*定义小按钮样式类*/
        font-size: 11px;
        padding: .2em 1em .275em;
    }
    /*设计颜色样式类：黑色风格的按钮*/
    /*通过设计不同颜色样式类，设计不同风格的按钮效果*/
    .black { /*黑色样式类*/
        color: #d7d7d7;
        border: solid 1px #333;
        background: #333;
        background: -webkit-gradient(linear, left top, left bottom, from(#666), to(#000));
        background: -moz-linear-gradient(top, #666, #000);
        background: linear-gradient(to top, #666, #000);
        filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#666666', endColorstr='#000000');
    }
    .black:hover { /*黑色鼠标经过样式类*/
        background: #000;
        background: -webkit-gradient(linear, left top, left bottom, from(#444), to(#000));
        background: -moz-linear-gradient(top, #444, #000);
        background: linear-gradient(to top, #444, #000);
        filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#444444', endColorstr='#000000');
    }
    .black:active { /*黑色激活样式类*/
        color: #666;
        background: -webkit-gradient(linear, left top, left bottom, from(#000), to(#444));
```



Note



```
        background: -moz-linear-gradient(top, #000, #444);
        background: linear-gradient(to top, #000, #444);
        filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#000000', endColorstr='#666666');}
</style>
<div>
        <a href="#" class="button black">圆角按钮</a>
        <a href="#" class="button black bigrounded">大号椭圆按钮</a>
        <a href="#" class="button black medium">中号按钮</a>
        <a href="#" class="button black small">小号按钮</a> <br />
</div>
```



本示例设计的按钮具有如下特点：

☑ 不需要图片和 JavaScript。

☑ 兼容 IE、Firefox 3.6+、Chrome 和 Safari 等主流浏览器。

☑ 支持 3 种按钮状态，如正常、悬停和激活。

☑ 可以应用到任何 HTML 元素，如 a、input、button、span、div、p、h3 等。

☑ 安全兼容不支持 CSS3 的浏览器，如果不兼容 CSS3，则显示没有渐变和阴影的普通按钮。

视频讲解

5.2.11 案例：设计图标

本例通过 CSS3 径向渐变制作圆形图标特效，设计效果如图 5.42 所示。在内部样式表中，使用 radial-gradient()函数为图标标签定义径向渐变背景，设计立体效果；使用“border-radius:50%;”声明定义图标显示为圆形；使用 box-shadow 属性为图标添加投影；使用 text-shadow 属性为图标文本定义润边效果；使用 radial-gradient 设计环形径向渐变效果，为图标添加高亮特效。



图 5.42　设计径向渐变图标效果

示例主要代码如下：

```
<style type="text/css">
.icon {
        /*固定大小，可根据实际需要酌情调整，调整时应同步调整 line-height:60px;*/
        width: 60px; height: 60px;
        /*行内块显示，统一图标显示属性*/
        display:inline-block;
        /*清除边框，避免边框对整体特效的破坏*/
        border: none;
        /*设计圆形效果*/
        border-radius: 50%;
        /*定义图标阴影，第一个外阴影设计立体效果，第二个内阴影设计高亮特效*/
        box-shadow: 0 1px 5px rgba(255,255,255,.5) inset,
                        0 -2px 5px rgba(0,0,0,.3) inset, 0 3px 8px rgba(0,0,0,.8);
        /*定义径向渐变，模拟明暗变化的表面效果*/
        background: -webkit-radial-gradient( circle at top center, #f28fb8, #e982ad, #ec568c);
        background: radial-gradient(circle at top center, #f28fb8, #e982ad, #ec568c);
        /*定义图标字体样式*/
        font-size: 32px;
        color: #dd5183;
        text-align:center;          /*文本水平居中显示*/
        line-height:60px;           /*文本垂直居中显示，必须与 height: 60px;保持一致*/
```



```
        /*为文本添加阴影，第一个阴影设计立体效果，第二个阴影定义高亮特效*/
        text-shadow: 0 3px 10px #f1a2c1,
                     0 -3px 10px #f1a2c1;
    }
    </style>
    <div class="icon">Dw</div>
    <span class="icon">Fl</span>
    <p class="icon">PS</p>
```



Note

5.3　案例实战

本节将通过多个较复杂案例练习背景样式的实际应用。

5.3.1　设计优惠券

视频讲解

本例使用径向渐变设计一张优惠券，效果如图 5.43 所示。

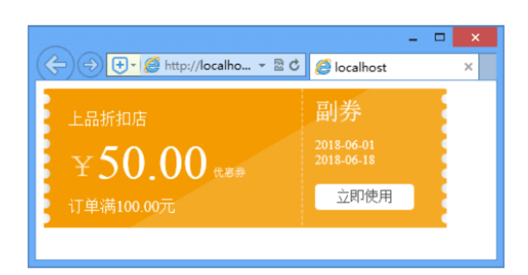


图 5.43　设计优惠券效果

先了解本案例的 HTML 结构：整个界面包裹在<div class="stamp stamp_yellow">标签中，stamp 类样式定制优惠券结构样式，stamp_yellow 类样式定制优惠券风格样式，即配色效果；在该包含框中，嵌入了两个子结构，<div class="par">负责设计左侧文本显示，<div class="copy">负责定制右侧信息；在包含框的底部嵌入一个<i>标签，该标签负责设计优惠券右下高亮显示面。

示例主要代码如下，样式代码解读请参考注释文本。

```
<style type="text/css">
/*通用类样式*/
.stamp {
    width: 387px; height: 140px;              /*固定大小，方便设计*/
    padding: 0 10px;                          /*左右留出 10px 空间，用来设计锯齿边沿效果*/
    position: relative;                       /*相对定位，定义定位包含框，方便内部对象定位显示*/
    overflow: hidden;                         /*禁止超出显示，避免破坏券面布局*/
}
.stamp:before {/*设计底色*/
    content: '';                              /*设计一个空的内容层*/
    position: absolute;                       /*绝对定位显示*/
    z-index: -1;                              /*让该层显示在文本的下面*/
    top: 0;                                   /*定义大小，顶部对齐*/
    bottom: 0;                                /*定义大小，底部对齐*/
    left: 10px;                               /*定义大小，左侧留白 10px*/
    right: 10px;                              /*定义大小，右侧留白 10px*/
}
.stamp:after {/*设计底色阴影*/
```





Note

```
        content: '';                             /*设计一个空的内容层*/
        position: absolute;                      /*绝对定位显示*/
        left: 10px;                              /*定义大小，左侧留白 10px*/
        top: 10px;                               /*定义大小，顶部留白 10px*/
        right: 10px;                             /*定义大小，右侧留白 10px*/
        bottom: 10px;                            /*定义大小，底部留白 10px*/
        box-shadow: 0 0 20px 1px rgba(0, 0, 0, 0.5);  /*为左右锯齿设计阴影效果*/
        z-index: -2;                             /*设计该层显示在最底部*/
    }
    .stamp i {/*设计高亮面*/
        position: absolute;                      /*绝对定位显示*/
        left: 20%; top: 45px;                    /*显示位置*/
        height: 190px; width: 390px;             /*定义大小*/
        background-color: rgba(255,255,255,.15); /*定义淡淡的高亮色*/
        transform: rotate(-30deg);               /*旋转角度，覆盖在右下面显示*/
    }
    .stamp .par {/*设计左侧文本样式*/
        float: left;
        padding: 16px 15px;
        width: 220px;
        border-right: 2px dashed rgba(255,255,255,.3); /*在正、副券绘制一条垂直虚线*/
        text-align: left;
    }
    .stamp .par p { color: #fff; margin:6px 0; }        /*设计正文文本样式*/
    .stamp .par span {                           /*设计金额样式*/
        font-size: 50px;
        color: #fff;
        margin-right: 5px;
    }
    .stamp .par .sign { font-size: 34px; }       /*设计人民币符号样式*/
    .stamp .par sub {
        position: relative;                      /*相对定位，方便移位*/
        top: -5px;                               /*下移，底部对齐*/
        color: rgba(255,255,255,.8);
    }
    .stamp .copy {                               /*设计右侧文本样式*/
        display: inline-block;
        padding: 21px 14px;
        width: 100px;
        vertical-align: text-bottom;
        font-size: 30px;
        color: rgb(255,255,255);
        padding: 10px 6px 10px 12px;
        font-size: 24px;
    }
    .stamp .copy p {
        font-size: 13px;
        margin-top: 12px;
        margin-bottom: 16px;
    }
    .stamp .copy a {                             /*设计扁平化按钮样式*/
```

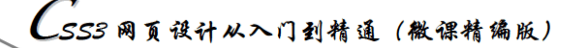


```
        background-color: #fff;
        color: #333;
        font-size: 14px;
        text-decoration: none;
        text-align:center;
        padding: 5px 10px;
        border-radius: 4px;
        display: block;
    }
    /*设计风格*/
    /*鹅黄*/
    .stamp_yellow   {/*正文背景样式，通过径向渐变定义圆形纹理背景*/
        background: #F39B00;
        background: radial-gradient(rgba(0, 0, 0, 0) 0, rgba(0, 0, 0, 0) 5px, #F39B00 5px);
        background-size: 15px 15px;            /*定义每个圆形大小*/
        background-position: 9px 3px;          /*左右两侧显示圆孔形背景*/
    }
    /*设计正文部分仅显示单色背景，左右边沿显示圆孔锯齿背景*/
    .stamp_yellow:before {background-color: #F39B00;}
    </style>
    <div class="stamp stamp_yellow">
        <div class="par">
            <p>上品折扣店</p>
            <sub class="sign">￥</sub><span>50.00</span><sub>优惠券</sub>
            <p>订单满 100.00 元</p>
        </div>
        <div class="copy">副券
            <p>2018-06-01<br>
                2018-06-18</p>
            <a href="#">立即使用</a></div>
        <i></i>
    </div>
```

在上面示例基础上，用户可以设计不同风格的界面效果。例如，重新定义主题风格样式，可以拓展更多主题的优惠券。主要技巧在于修改正文背景色和径向渐变的第二个颜色（test2.html），效果如图 5.44 所示。

```
    /*浅红*/
    .stamp_red {
        background: #D24161;
        background: radial-gradient(transparent 0, transparent 5px, #D24161 5px);
        background-size: 15px 15px;
        background-position: 9px 3px;
    }
    .stamp_red:before {background-color: #D24161;}
    /*浅绿*/
    .stamp_green {
        background: #7EAB1E;
        background: radial-gradient(transparent 0, transparent 5px, #7EAB1E 5px);
        background-size: 15px 15px;
```

Note





Note

```
        background-position: 9px 3px;
}
.stamp_green:before {background-color: #7EAB1E;}
/*天蓝*/
.stamp_blue {
        width: 390px;
        background: #50ADD3;
        background: radial-gradient(rgba(0, 0, 0, 0) 0, rgba(0, 0, 0, 0) 4px, #50ADD3 4px);
        background-size: 12px 8px;
        background-position: -5px 10px;
}
.stamp_blue:before {background-color: #50ADD3;}
```



图 5.44　设计不同风格的优惠券效果



视 频 讲 解

5.3.2　设计桌面纹理背景

本例使用 CSS3 线性渐变属性制作纹理图案，主要利用多重背景进行设计，然后使用线性渐变绘制每一条线，通过叠加和平铺，完成重复性纹理背景效果，如图 5.45 所示。

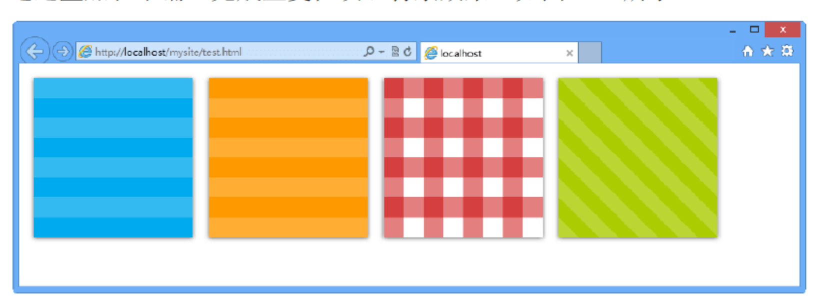


图 5.45　定义网页纹理背景效果

案例主要代码如下：

```
<style type="text/css">
.patterns {
        width: 200px; height: 200px; float: left; margin: 10px;
        box-shadow: 0 1px 8px #666;}
.pt1 {
        background-size: 50px 50px;
        background-color: #0ae;
```



```
        background-image: -webkit-linear-gradient(rgba(255, 255, 255, .2) 50%, transparent 50%, transparent);
        background-image: linear-gradient(rgba(255, 255, 255, .2) 50%, transparent 50%, transparent);}
    .pt2 {
        background-size: 50px 50px;
        background-color: #f90;
        background-image: -webkit-linear-gradient(0deg, rgba(255, 255, 255, .2) 50%, transparent 50%, transparent);
        background-image: linear-gradient(0deg, rgba(255, 255, 255, .2) 50%, transparent 50%, transparent);}
    .pt3 {
        background-size: 50px 50px;
        background-color: white;
        background-image: -webkit-linear-gradient(to top, transparent 50%, rgba(200, 0, 0, .5) 50%, rgba(200, 0,
0, .5)),   -webkit-linear-gradient(to left, transparent 50%, rgba(200, 0, 0, .5) 50%, rgba(200, 0, 0, .5));
        background-image: linear-gradient(to top, transparent 50%, rgba(200, 0, 0, .5) 50%, rgba(200, 0, 0, .5)),
linear-gradient(to left, transparent 50%, rgba(200, 0, 0, .5) 50%, rgba(200, 0, 0, .5))}
    .pt4 {
        background-size: 50px 50px;
        background-color: #ac0;
        background-image: -webkit-linear-gradient(45deg, rgba(255, 255, 255, .2) 25%, transparent 25%, transparent
50%, rgba(255, 255, 255, .2) 50%, rgba(255, 255, 255, .2) 75%, transparent 75%, transparent);
        background-image: linear-gradient(45deg, rgba(255, 255, 255, .2) 25%, transparent 25%, transparent 50%,
rgba(255, 255, 255, .2) 50%, rgba(255, 255, 255, .2) 75%, transparent 75%, transparent);}
    </style>
    </head>
    <body>
    <div class="patterns pt1"></div>
    <div class="patterns pt2"></div>
    <div class="patterns pt3"></div>
    <div class="patterns pt4"></div>
    </body>
    </html>
```

灵活使用径向渐变和线性渐变，用户还可以设计更多图案（test2.html），如图 5.46 所示。

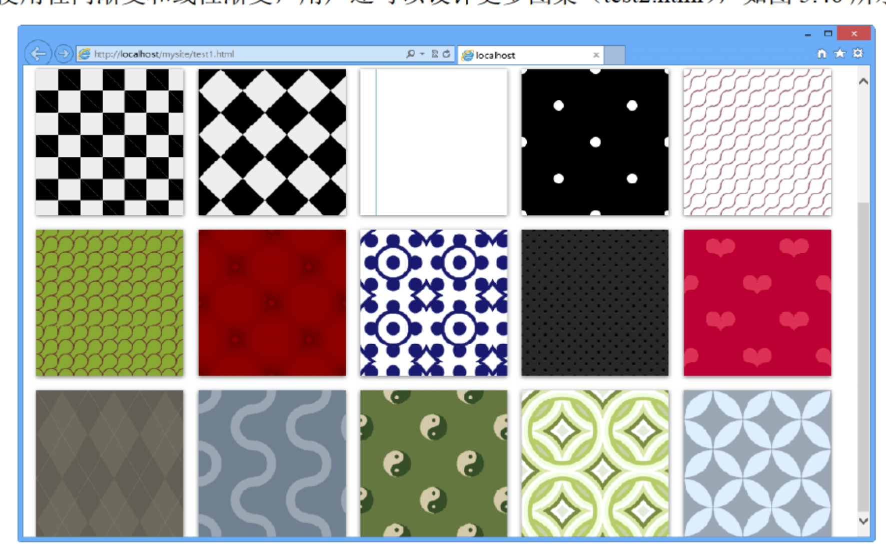


图 5.46　设计丰富的纹理背景效果



视频讲解



Note

5.3.3 渐变特殊应用场景

渐变可以用在 border-image-source、background-image、list-style-image、cursor 等属性上，用来取代 url 属性值。前面各节主要针对 background-image 属性进行介绍，下面结合示例介绍其他属性的应用情形。

1．定义渐变效果的边框

【示例 1】本例通过 CSS3 渐变，为 border-image 属性定义渐变边框，效果如图 5.47 所示。

```
<style type="text/css">
div {
    width: 400px;
    height: 200px;
    margin: 20px;
    border: solid #000 50px;
    -webkit-border-image:-webkit-linear-gradient(yellow, blue 20%, #0f0) 50;    /*Safari 5.1- 6.0*/
    -o-border-image: -o-linear-gradient(yellow, blue 20%, #0f0) 50;             /*Opera 11.1 - 12.0*/
    -moz-border-image: -moz-linear-gradient(yellow, blue 20%, #0f0) 50;         /*Firefox 3.6 - 15*/
    border-image: linear-gradient(yellow, blue 20%, #0f0) 50;                   /*标准语法*/
}
</style>
<div></div>
```

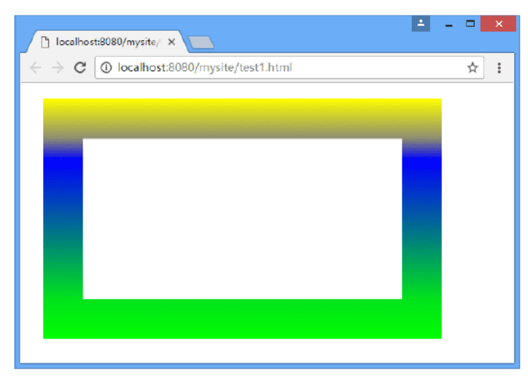


图 5.47　设计渐变边框效果

2．定义填充内容效果

【示例 2】本例通过 conten 属性，为<div class="div1">标签嵌入一个通过渐变设计的圆球，同时为这个包含框设计一个渐变背景，从而产生一种透视框的效果，如图 5.48 所示（test1.html）。

```
<style type="text/css">
.div1 { /*设计包含框的外形和大小*/
    width: 400px; height: 200px;
    border: 20px solid #A7D30C;
}
.div1:before {
    /*在 div 元素前插入内容对象，在该对象中绘制一个背景图形，并定义显示边框效果*/
    /*Safari 5.1 - 6.0*/
```



```
        content: -webkit-radial-gradient(left bottom, farthest-side, #f00, #f99 60px, #005);
        /*Opera 11.6 - 12.0*/
        content: -o-radial-gradient(left bottom, farthest-side, #f00, #f99 60px, #005);
        /*Firefox 3.6 - 15*/
        content: -moz-radial-gradient(left bottom, farthest-side, #f00, #f99 60px, #005);
        /*标准语法*/
        content: radial-gradient(farthest-side at left bottom, #f00, #f99 60px, #005);
}
</style>
<div class="div1"></div>
```



3．定义列表图标

【示例 3】本例通过 list-style-image 属性，为 ul 元素定义自定义图标，该图标通过渐变特效进行绘制，从而产生一种精致的二色效果，如图 5.49 所示。

```
<style type="text/css">
ul {list-style-image: linear-gradient(red 50%, blue 50%);}
</style>
<ul>
        <li>HTML5</li>
        <li>CSS3</li>
        <li>JavaScript</li>
</ul>
```

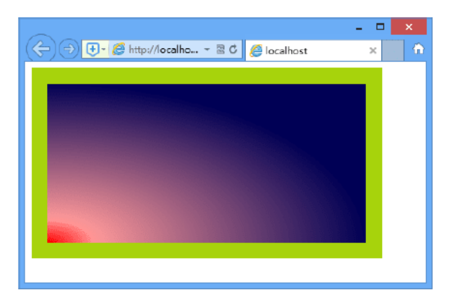


图 5.48　插入球形内容填充物并显示边框效果

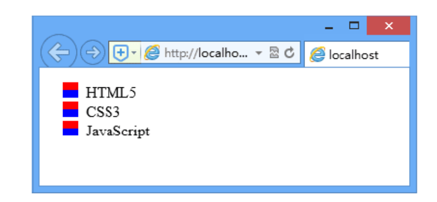


图 5.49　设计项目符号效果

5.3.4　设计栏目折角效果



灵活使用 CSS3 渐变背景，可以创作出很多新颖的设计。

【示例 1】使用线性渐变设计右上角缺角的栏目，效果如图 5.50 所示。

```
<style type="text/css">
.box {
        background: linear-gradient(-135deg, transparent 30px, #162e48 30px);
        color: #fff;
        padding: 12px 24px;
}
</style>
<div class="box">
        <h1>W3C 发布 HTML5 的正式推荐标准</h1>
        <p>2014 年 10 月 28 日，W3C 的 HTML 工作组正式发布了 HTML5 的正式推荐标准（W3C Recommendation）。
```



```
W3C 在美国圣克拉拉举行的 W3C 技术大会及顾问委员会会议（TPAC 2014）上宣布了这一消息。 HTML5 是万维网的核心语言——可扩展标记语言的第 5 版。在这一版本中，增加了支持 Web 应用开发者的许多新特性，以及更符合开发者使用习惯的新元素，并重点关注定义清晰的、一致的准则，以确保 Web 应用和内容在不同用户代理（浏览器）中的互操作性。HTML5 是构建开放 Web 平台的核心。</p>
        <p class="right">更多<a href="http://www.chinaw3c.org/archives/677/" target="_blank">详细内容</a></p>
    </div>
```



Note

【示例 2】使用线性渐变设计右上角补角的栏目，效果如图 5.51 所示。

```
<style type="text/css">
.box {
    background: linear-gradient(-135deg, #f00 30px, #fff 30px, #162e48 32px);
    color: #fff;
    padding: 12px 24px;
}
</style>
```

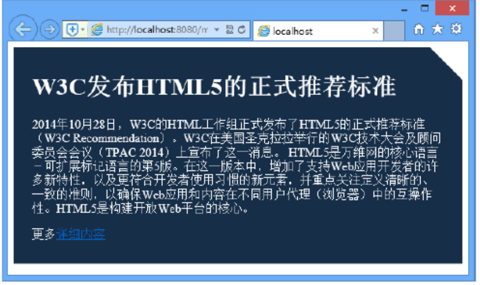


图 5.50　设计缺角栏目效果

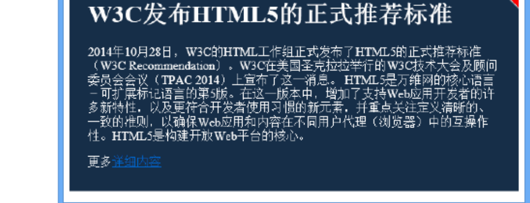


图 5.51　设计补角栏目效果

【示例 3】使用 box-shadow 为栏目加上高亮边框，同时需要设计网页背景色为深色，效果如图 5.52 所示。

```
<style type="text/css">
body {background:#000;}
.box {
    background: linear-gradient(-135deg, #f00 30px, #fff 30px, #162e48 32px);
    color: #fff;
    padding: 12px 24px;
    box-shadow: 0 0 1px 1px #fff inset;
}
</style>
```

【示例 4】我们无法直接为缺角栏目设计边框线效果，考虑到兼容性问题，可以使用:before 和:after 实现该效果。注意，网页背景色为深色，与.box:after 边框色保持一致，如图 5.53 所示。

```
<style type="text/css">
body {background: #000;}
.box {
    background: #162e48;
    color: #fff;
    padding: 12px 24px;
    position: relative;
    border: 1px solid  #fff;
```



```
}
.box:before {
    content: ' ';
    border: solid transparent;
    position: absolute;
    border-width: 30px;
    border-top-color: #fff;
    border-right-color: #fff;
    right: 0px;
    top: 0px;
}
.box:after {
    content: ' ';
    border: solid transparent;
    position: absolute;
    border-width: 30px;
    border-top-color: #000;
    border-right-color: #000;
    top: -1px;
    right: -1px;
}
</style>
```

图 5.52　设计高亮边框栏目效果



图 5.53　设计缺角边框栏目效果

【代码解析】

第 1 步，在示例 4 中，首先使用.box:before 在容器内容前面插入一个粗边框对象：白色边框，宽度为 30px，由于内容为空 content: ' ';，则收缩为一团，显示如图 5.54 所示。

第 2 步，使用绝对定位，将三角填充物精确定位到右上角显示，如图 5.55 所示。



图 5.54　设计三角填充物



图 5.55　定位三角填充物到栏目右上角





第 3 步，使用.box:after 在栏目内容的后面插入一个同样大小的三角形填充物，边框色为背景色，即黑色，如图 5.56 所示。



图 5.56　插入一个黑色三角填充物

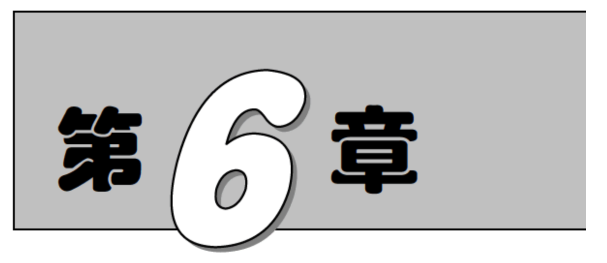
第 4 步，使用绝对定位，将黑色三角填充物精确定位到右上角显示，并向右上角偏移 1px，遮盖住白色区域，留一条白色缝隙，即可完成本例效果设计。

5.4　在 线 练 习

本节练习使用 CSS3 设计各种网页图像效果和背景图像特效，强化基本功训练。



在 线 练 习



使用 CSS3 美化列表和超链接样式

在默认状态下，超链接文本显示为蓝色且带有下画线，当鼠标指针移过链接对象时显示为手形，访问过的超链接文本显示为紫色；列表项目缩进显示，并在左侧显示项目符号。在网页设计过程中，一般都会根据个人喜好和实际需要重新定义超链接和列表样式。

【学习重点】

- 正确使用行为伪类。
- 能够灵活设计符合页面风格的链接样式。
- 定义列表样式。
- 能够根据页面风格设计列表菜单样式。



6.1 设计超链接样式



Note



视频讲解

下面介绍如何使用 CSS3 设计超链接的基本样式。

6.1.1 使用动态伪类

在网页设计中，用户可以使用 CSS3 的动态伪类选择器定义超链接的 4 种状态样式。

☑ a:link：定义超链接的默认样式。

☑ a:visited：定义超链接被访问后的样式。

☑ a:hover：定义鼠标指针移过超链接时的样式。

☑ a:active：定义超链接被激活时的样式。

【示例】在下面示例中，定义页面所有超链接默认为红色、下画线效果，当鼠标经过时显示为绿色、下画线效果，而当被单击时则显示为黄色、下画线效果，被访问过之后显示为蓝色、下画线效果，演示效果如图 6.1 所示。

```
<style type="text/css">
a:link {color: #FF0000;/*红色*/} /*超链接默认样式*/
a:visited {color: #0000FF; /*蓝色*/} /*超链接被访问后的样式*/
a:hover {color: #00FF00; /*绿色*/}/*鼠标经过超链接的样式*/
a:active {color: #FFFF00; /*黄色*/}/*超链接被激活时的样式*/
</style>
<ul class="p1">
    <li><a href="#" class="a1">首页</a></li>
    <li><a href="#" class="a2">新闻</a></li>
    <li><a href="#" class="a3">微博</a></li>
</ul>
<ul class="p2">
    <li><a href="#" class="a1">关于</a></li>
    <li><a href="#" class="a2">版权</a></li>
    <li><a href="#" class="a3">友情链接</a></li>
</ul>
```

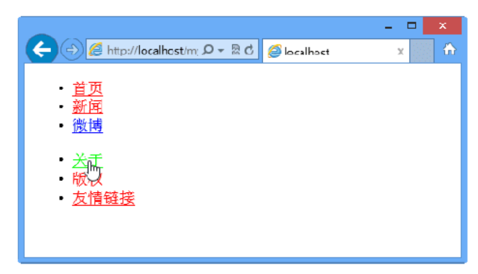


图 6.1 定义超链接样式

提示：超链接 4 种状态样式的排列顺序是固定的，一般不能随意调换。正确顺序是：link、visited、hover 和 active。



在下面样式中，当鼠标经过超链接时，会先执行第 1 行声明，但是紧接着第 3 行的声明会覆盖掉第 1 行和第 2 行声明的样式，所以就无法看到鼠标经过和被激活时的效果。

```
a.a1:hover {color: #00FF00;}
a.a1:active {color: #FFFF00;}
a.a1:link {color: #FF0000;}
a.a1:visited {color: #0000FF;}
```



在上面代码中，通过指定类型选择器，限定上面 4 个样式仅作用于包含 a1 类的超链接中。

当然，用户可以根据需要仅定义部分状态样式。例如，若要把未访问的和已经访问的链接定义成相同的样式，则可以定义 link、hover 和 active 3 种状态。

```
a.a1:link {color: #FF0000;}
a.a1:hover {color: #00FF00;}
a.a1:active {color: #FFFF00;}
```

如果仅希望超链接显示两种状态样式，可以使用 a 和 hover 来定义。其中 a 标签选择器定义 a 元素的默认显示样式，然后定义鼠标经过时的样式。

```
a {color: #FF0000;}
a:hover {color: #00FF00;}
```

但是如果页面中包含锚记对象，将会影响锚记的样式。如果定义如下的样式，则仅影响超链接未访问时的样式和鼠标经过时的样式。

```
a:link {color: #FF0000;}
a:hover {color: #00FF00;}
```

6.1.2　定义下画线样式



视频讲解

在设计超链接样式时，下画线一直是一个重要效果，巧妙结合下画线、边框和背景图像，可以设计出很多富有个性的样式。例如，定义下画线的色彩、距离、长度和对齐方式，以及定制双下画线等。

如果用户不喜欢超链接文本的下画线样式，可以使用 CSS3 的 text-decoration 属性进行清除。

```
a {/*完全清除超链接的下画线效果*/
    text-decoration:none;
}
```

从用户体验的角度考虑，在取消默认的下画线之后，应确保浏览者可以识别所有超链接，如加粗显示、变色、缩放、高亮背景等。也可以设计当鼠标经过时增加下画线，因为下画线具有很好的提示作用。

```
a:hover {/*鼠标经过时显示下画线效果*/
    text-decoration:underline;
}
```

下画线样式不仅仅是一条实线，可以根据需要自定义设计。主要设计思路如下：

☑　借助<a>标签的底边框线来实现。

☑　利用背景图像来实现，背景图像可以设计出更多精巧的下画线样式。

【示例 1】下面示例设计当鼠标经过超链接文本时，显示为下画虚线、字体加粗、色彩高亮的效果，如图 6.2 所示。

```
<style type="text/css">
a {/*超链接的默认样式*/
```



Note

```
        text-decoration:none;                       /*清除超链接下画线*/
        color:#999;                                 /*浅灰色文字效果*/
    }
    a:hover {/*鼠标经过时样式*/
        border-bottom:dashed 1px red;               /*鼠标经过时显示虚下画线效果*/
        color:#000;                                 /*加重颜色显示*/
        font-weight:bold;                           /*加粗字体显示*/
        zoom:1;                                     /*解决 IE 浏览器无法显示问题*/
    }
    </style>
    <ul class="p1">
        <li><a href="#" class="a1">首页</a></li>
        <li><a href="#" class="a2">新闻</a></li>
        <li><a href="#" class="a3">微博</a></li>
    </ul>
```

【示例 2】可以使用 CSS3 的 border-bottom 属性定义超链接文本的下画线样式。下面示例定义超链接始终显示为下画线效果，并通过颜色变化来提示鼠标经过时的状态变化，效果如图 6.3 所示。

```
    <style type="text/css">
    a {/*超链接的默认样式*/
        text-decoration:none;                       /*清除超链接下画线*/
        border-bottom:dashed 1px red;               /*红色虚下画线效果*/
        color:#666;                                 /*灰色字体效果*/
        zoom:1;                                     /*解决 IE 浏览器无法显示问题*/
    }
    a:hover {/*鼠标经过时样式*/
        color:#000;                                 /*加重颜色显示*/
        border-bottom:dashed 1px #000;              /*改变虚下画线的颜色*/
    }
    </style>
```

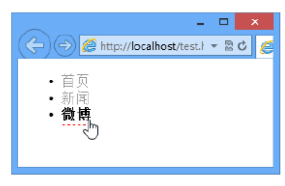


图 6.2　定义下画线样式（1）

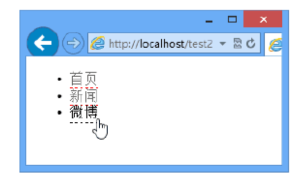


图 6.3　定义下画线样式（2）

【示例 3】使用 CSS3 的 background 属性可以借助背景图定义更精致、个性的下画线样式。

【操作步骤】

第 1 步，使用 Photoshop 设计一个虚线（images/dashed.psd），设计图像高度为 1px，宽度为 4px、6px 或 8px。具体宽度可根据虚线的疏密确定。

第 2 步，在 Photoshop 中，选择颜色以跳格方式进行填充，最后保存为 GIF 格式图像。

第 3 步，把示例 2 另存为 test3.html，使用背景图代替 border-bottom:dashed 1px red;声明，主要样式代码如下：

```
    <style type="text/css">
    a {/*超链接的默认样式*/
```



```
    text-decoration:none;                    /*清除超链接下画线*/
    color:#666;                              /*灰色字体效果*/
}
a:hover {/*鼠标经过时样式*/
    color:#000;                              /*加重颜色显示*/
    /*定义背景图像，定位到超链接元素的底部，并沿 x 轴水平平铺*/
    background:url(images/dashed1.gif) left bottom repeat-x;
}
</style>
```



第 4 步，在浏览器中预览，效果如图 6.4 所示。

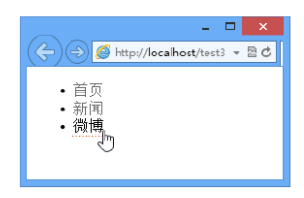


图 6.4　背景图像设计的下画线样式

6.1.3　定义特效样式



视 频 讲 解

本节通过示例介绍如何为超链接文本设计立体视觉效果，主要是借助边框颜色的深浅错落，模拟一种凸凹变化的立体效果。设计技巧如下：

☑　设置右边框和底边框同色，同时设置顶边框和左边框同色，利用明暗色彩的搭配来设计立体效果。

☑　设置超链接文本的背景色为深色效果，营造凸起效果，当鼠标移过时，再定义浅色背景来营造凹下效果。

☑　为网页设计浅色背景，再定义字体颜色来烘托立体样式。

【示例】在这个示例中，定义超链接在默认状态下显示灰色右边和底边框线效果、白色顶边和左边框线效果。而当鼠标移过时，则清除右侧和底部边框线，并定义左侧和顶部边框效果，演示效果如图 6.5 所示。

```
<style type="text/css">
body {background:#fcc;} /*浅色网页背景*/
ul {list-style-type: none;} /*清除项目符号*/
li {margin: 0 2px; float: left;} /*并列显示*/
a {/*超链接的默认样式*/
    text-decoration:none;                    /*清除超链接下画线*/
    border:solid 1px;                        /*定义 1px 实线边框*/
    padding: 0.4em 0.8em;                    /*增加超链接补白*/
    color: #444;                             /*定义灰色字体*/
    background: #f99;                        /*超链接背景色*/
    border-color: #fff #aaab9c #aaab9c #fff; /*分配边框颜色*/
    zoom:1;                                  /*解决 IE 浏览器无法显示问题*/
}
a:hover {/*鼠标经过时样式*/
    color: #800000;                          /*超链接字体颜色*/
    background: transparent;                 /*清除超链接背景色*/
```



```
        border-color: #aaab9c #fff #fff #aaab9c;                /*分配边框颜色*/
    }
    </style>
```



图 6.5　定义立体样式



Note



视频讲解

6.1.4　定义光标样式

在默认状态下，鼠标指针经过超链接时显示为手形。使用 CSS 的 cursor 属性可以改变这种默认效果，cursor 属性定义鼠标移过对象时的指针样式，取值说明如表 6.1 所示。

表 6.1　cursor 属性取值说明

取　　值	说　　明
auto	基于上下文决定应该显示什么光标
crosshair	十字线光标（+）
default	基于平台的默认光标。通常渲染为一个箭头
pointer	指针光标，表示一个超链接
move	十字箭头光标，用于标示对象可被移动
e-resize、ne-resize、nw-resize、n-resize、se-resize、sw-resize、s-resize、w-resize	表示正在移动某个边，如 se-resize 光标用来表示框的移动开始于东南角
text	表示可以选择文本。通常渲染为 I 形光标
wait	表示程序正忙，需要用户等待，通常渲染为手表或沙漏
help	光标下的对象包含帮助内容，通常渲染为一个问号或一个气球
<uri>URL	自定义光标类型的图标路径

如果自定义光标样式。使用绝对或相对 URL 指定光标文件（后缀为.cur 或者.ani）。

【示例】下面示例在内部样式表中定义多个鼠标指针类样式，然后为表格单元格应用不同的类样式，完整代码可以参考本节示例源代码，示例演示效果如图 6.6 所示。

```
<style>
.auto {cursor: auto;}
.default {cursor: default;}
.none {cursor: none;}
.context-menu {cursor: context-menu;}
.help {cursor: help;}
.pointer {cursor: pointer;}
.progress {cursor: progress;}
.wait {cursor: wait;}
.cell {cursor: cell;}
.crosshair {cursor: crosshair;}
.text {cursor: text;}
.vertical-text {cursor: vertical-text;}
.alias {cursor: alias;}
.copy {cursor: copy;}
```



```
.move {cursor: move;}
.no-drop {cursor: no-drop;}
.not-allowed {cursor: not-allowed;}
.e-resize {cursor: e-resize;}
.n-resize {cursor: n-resize;}
.ne-resize {cursor: ne-resize;}
.nw-resize {cursor: nw-resize;}
.s-resize {cursor: s-resize;}
.se-resize {cursor: se-resize;}
.sw-resize {cursor: sw-resize;}
.w-resize {cursor: w-resize;}
.ew-resize {cursor: ew-resize;}
.ns-resize {cursor: ns-resize;}
.nesw-resize {cursor: nesw-resize;}
.nwse-resize {cursor: nwse-resize;}
.col-resize {cursor: col-resize;}
.row-resize {cursor: row-resize;}
.all-scroll {cursor: all-scroll;}
.zoom-in {cursor: zoom-in;}
.zoom-out {cursor: zoom-out;}
.url {cursor: url(skin/cursor.gif), url(skin/cursor.png), url(skin/cursor.jpg), pointer;}
</style>
```



图 6.6　比较不同光标样式效果

提示：使用自定义图像作为光标类型，IE 和 Opera 只支持*.cur 等特定的图片格式；而 Firefox、Chrome 和 Safari 既支持特定图片类型，也支持常见的*.jpg、*.gif、*.jpg 等图片格式。cursor 属性值可以是一个序列，当用户端无法处理第 1 个图标时，它会尝试处理第 2 个、第 3 个等，如果用户端无法处理任何定义的光标，则必须使用列表最后的通用光标。例如，下面样式中就定义了 3 个自定义动画光标文件，最后定义了 1 个通用光标类型。

```
a:hover {cursor:url('images/1.ani'), url('images/1.cur'), url('images/1.gif'), pointer;}
```

6.2　设计列表样式

下面介绍如何使用 CSS3 设计列表的基本样式。


视频讲解


Note

6.2.1 定义项目符号类型

使用 CSS3 的 list-style-type 属性可以定义列表项目符号的类型，也可以取消项目符号，该属性取值说明如表 6.2 所示。

表 6.2 list-style-type 属性值

属 性 值	说 明	属 性 值	说 明
disc	实心圆，默认值	upper-roman	大写罗马数字
circle	空心圆	lower-alpha	小写英文字母
square	实心方块	upper-alpha	大写英文字母
decimal	阿拉伯数字	none	不使用项目符号
lower-roman	小写罗马数字	armenian	传统的亚美尼亚数字
cjk-ideographic	浅白的表意数字	georgian	传统的乔治数字
lower-greek	基本的希腊小写字母	hebrew	传统的希伯来数字
hiragana	日文平假名字符	hiragana-iroha	日文平假名序号
katakana	日文片假名字符	katakana-iroha	日文片假名序号
lower-latin	小写拉丁字母	upper-latin	大写拉丁字母

使用 CSS3 的 list-style-position 属性可以定义项目符号的显示位置。该属性取值包括 outside 和 inside，其中 outside 表示把项目符号显示在列表项的文本行以外，列表符号默认显示为 outside；inside 表示把项目符号显示在列表项文本行以内。

注意： 如果要清除列表项目的缩进显示样式，可以使用下面样式实现:

```
ul, ol {
    padding: 0;
    margin: 0;
}
```

【示例】 下面示例定义项目符号显示为空心圆，并位于列表行内部，如图 6.7 所示。

```
<style type="text/css">
body {/*清除页边距*/
    margin: 0;                          /*清除边界*/
    padding: 0;                         /*清除补白*/
}
ul {/*列表基本样式*/
    list-style-type: circle;            /*空心圆符号*/
    list-style-position: inside;        /*显示在里面*/
}
</style>
<ul>
    <li><a href="#">关于我们</a></li>
    <li><a href="#">版权信息</a></li>
    <li><a href="#">友情链接</a></li>
</ul>
```



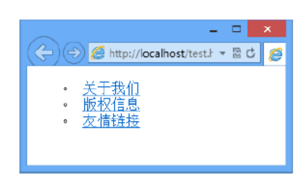


图 6.7　定义列表项目符号



> 提示：在定义列表项目符号样式时，应注意以下两点：
>
> （1）不同浏览器对于项目符号的解析效果及其显示位置略有不同。如果要兼容不同浏览器的显示效果，应关注这些差异。
>
> （2）项目符号显示在里面和外面会影响项目符号与列表文本之间的距离，同时影响列表项的缩进效果。不同浏览器在解析时会存在差异。

6.2.2　定义项目符号图像



使用 CSS3 的 list-style-image 属性可以自定义项目符号。该属性允许指定一个外部图标文件，以此满足个性化设计需求。用法如下：

```
list-style-image: none | <url>
```

默认值为 none。

【示例】以 6.2.1 节示例为基础，重新设计内部样式表，增加自定义项目符号，设计项目符号为外部图标 bullet_main_02.gif，效果如图 6.8 所示。

```
<style type="text/css">
ul {/*列表基本样式*/
    list-style-type: circle;                            /*空心圆符号*/
    list-style-position: inside;                        /*显示在里面*/
    list-style-image: url(images/bullet_main_02.gif);   /*自定义列表项目符号*/
}
</style>
```

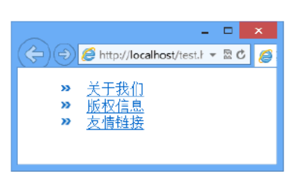


图 6.8　自定义列表项目符号

> 提示：当同时定义项目符号类型和自定义项目符号时，自定义项目符号将覆盖默认的符号类型。但是如果 list-style-type 属性值为 none 或指定的外部图标文件不存在时，则 list-style-type 属性值有效。

6.2.3　模拟项目符号



使用 CSS3 的 background 属性也可以模拟列表项目的符号，设计技巧如下：



（1）使用 list-style-type:none 隐藏列表的默认项目符号。

（2）使用 background 属性为列表项目定义背景图像，精确定位其显示位置。

（3）使用 padding-left 属性为列表项目定义左侧空白，避免背景图被项目文本遮盖。



Note

【示例】在下面示例中，先清除列表的默认项目符号，然后为项目列表定义背景图像，并定位到左侧垂直居中的位置，为了避免列表文本覆盖背景图像，定义左侧补白为一个字符宽度，这样就可以把列表信息向右缩进显示，显示效果如图 6.9 所示。

```
<style type="text/css">
ul {/*清除列表的默认样式*/
    list-style-type: none;
    padding: 0;
    margin: 0;
}
li {/*定义列表项目的样式*/
    background-image: url(images/bullet_sarrow.gif);            /*定义背景图像*/
    background-position: left center;                          /*精确定位背景图像的位置*/
    background-repeat: no-repeat;                              /*禁止背景图像平铺显示*/
    padding-left: 1em;                                         /*为背景图像挤出空白区域*/
}
</style>
```

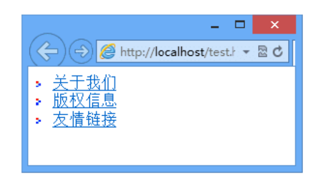


图 6.9　使用背景图模拟项目符号

6.3　案例实战

下面通过多个案例演示如何在具体页面中设计超链接和列表的样式。



6.3.1　设计图形按钮链接

超链接可以显示为多种样式，如动画、按钮、图像、特效等，本节介绍如何设计图形化按钮样式。这可以使用 CSS 的 background-image 属性实现。

【示例 1】下面示例用背景图像替换超链接文本，设计图形按钮效果，如图 6.10 所示。

```
<style type="text/css">
a.reg {/*超链接样式*/
    background: transparent url('images/btn2.gif') no-repeat top left;  /*背景图像*/
    display: block;                                            /*块状显示，方便定义宽度和高度*/
    width:74px;                                                /*宽度，与背景图像同宽*/
    height: 25px;                                              /*高度，与背景图像同高*/
    text-indent:-999px;                                        /*隐藏超链接中的文本*/
}
```



```
</style>

<a class="reg"   href="#">注册</a>
```



在上面代码中，使用 background-repeat 防止背景图重复平铺；定义<a>标签以块状或者行内块状显示，以方便为超链接定义高和宽；在定义超链接的显示大小时，其宽和高最好与背景图像保持一致，也可以使用 padding 属性撑开<a>标签，以代替 width 和 height 属性声明；使用 text-indent 属性隐藏超链接中的文本。

注意：如果超链接区域比背景图大，可以使用 background-position 属性定位背景图像在超链接中的显示位置。

【示例 2】下面示例为超链接不同状态定义不同背景图像：在正常状态下，超链接左侧显示一个箭头式的背景图像；当鼠标移过超链接时，背景图像被替换为另一个动态 GIF 图像，使整个超链接动态效果立即显示出来，演示效果如图 6.11 所示。

```
<style type="text/css">
a.reg {/*定义超链接正常样式：定位左侧背景图像*/
    background: url("images/arrow2.gif") no-repeat left center;
    padding-left:14px;
}
a.reg:hover {/*定义鼠标经过时超链接样式：定位左侧背景图像*/
    background: url("images/arrow1.gif") no-repeat left center;
    padding-left:14px;
}
</style>
<a class="reg"   href="#">注册</a>
```

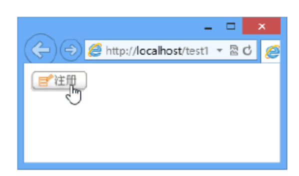


图 6.10　图形化按钮样式

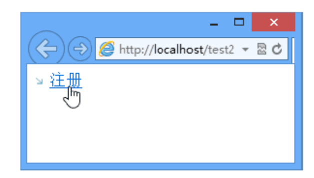


图 6.11　动态背景样式

在上面代码中，通过 padding-left 属性定义超链接左侧空隙，这样就可以使定义的背景图显示出来，避免被链接文本所遮盖。实战中，经常需要使用 padding 属性来为超链接增加空余的空间，以便背景图像能够很好地显示出来。

6.3.2　设计背景滑动样式



使用 CSS 滑动门技术可以设计宽度可伸缩的超链接样式。所谓滑动门，就是通过两个背景图像的叠加，来创造一些可自由伸缩的背景效果。

【操作步骤】

第 1 步，使用 Photoshop 设计好按钮图形的效果图，然后分切为两截，其中一截应尽可能短，只包括一条椭圆边，另一截可以尽可能长，这样设计的图形按钮就可以容纳更多的字符，如图 6.12 所示。

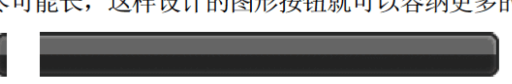


图 6.12　绘制并裁切滑动门背景图



第 2 步，启动 Dreamweaver，新建网页，保存为 test.html，在<body>标签内输入以下代码，构建一个可以定义重叠背景图的超链接结构，在每个超链接<a>标签中包含一个<span>辅助标签。



Note

```
<a  href="#"><span>按钮</span></a>
<a  href="#"><span>超链接</span></a>
<a  href="#"><span>图像按钮</span></a>
<a  href="#"><span>扩展性按钮</span></a>
<a  href="#"><span>能够定义很多字数的文本链接</span></a>
```

第 3 步，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后输入下面样式。使用 CSS 把短的背景图（left1.gif）固定在<a>标签的左侧。

```
a {/*定义超链接样式*/
    background: url('images/left1.gif') no-repeat top left;    /*把短截背景图像固定在左侧*/
    display: block;                                             /*以块状显示，这样能够定义大小*/
    float:left;                 /*浮动显示，这样 a 元素能够自动收缩宽度，以正好包容文本*/
    padding-left: 8px;          /*增加左侧内边距，该宽度正好与上面定义的背景图像同宽*/
    font: bold 13px Arial;                                      /*超链接文本字体属性*/
    line-height: 22px;                                          /*定义行高*/
    height: 30px;                                               /*定义按钮高度*/
    color: white;                                               /*字体颜色*/
    margin-left:6px;                                            /*左侧外边框*/
    text-decoration:none;                                       /*清除默认的下画线样式*/
}
```

第 4 步，把长的背景图（right1.gif）固定在<span>标签的右侧。

```
a span {
    background: url('images/right1.gif') no-repeat top right;  /*定义长截背景图像*/
    display: block;                                             /*块状显示*/
    padding: 4px 10px 4px 2px;                                  /*增加内边距*/
}
```

第 5 步，在浏览器中预览，显示效果如图 6.13 所示。如果希望鼠标经过时背景图像的色彩稍稍有点变化，以增加按钮的动态感，不妨在鼠标经过时增加一个下画线效果。

```
a:hover {text-decoration: underline;}
```



图 6.13　设计滑动门链接效果



视 频 讲 解

6.3.3　设计背景交换样式

本例设计两幅大小相同、效果不同的背景图像，然后使用 CSS 进行轮换显示，设计一种简单的鼠标动画效果。

【操作步骤】

第 1 步，使用 Photoshop 设计两幅大小相同，但效果略不同的图像，如图 6.14 所示。





bg1.gif　　bg2.gif

图 6.14　设计背景图像

第 2 步，启动 Dreamweaver，新建网页，保存为 test1.html，在<body>标签内输入以下代码，构建一个列表结构。

```
<ul>
    <li><a href="#">首页</a></li>
    <li><a href="#">新闻</a></li>
    <li><a href="#">微博</a></li>
</ul>
```

第 3 步，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后输入下面样式。

```
a {/*超链接的默认样式*/
    text-decoration:none;                              /*清除默认的下画线*/
    display:inline-block;                              /*行内块状显示*/
    padding:2px 1em;                                   /*为文本添加补白效果*/
    height:28px;                                       /*固定高度*/
    line-height:32px;                                  /*行高等于高度，设计垂直居中*/
    text-align:center;                                 /*文本水平居中*/
    background:url(images/b1.gif) no-repeat center;    /*定义背景图像 1，禁止平铺，居中*/
    color:#ccc;                                        /*浅灰色字体*/
}
a:hover {/*鼠标经过时样式*/
    background:url(images/b2.gif) no-repeat center;    /*定义背景图像 2，禁止平铺，居中*/
    color:#fff;                                        /*白色字体*/
}
```

在上面样式代码中，先定义超链接以行内块状显示，这样便于控制它的宽和高，然后根据背景图像大小定义 a 元素的大小，并分别为默认状态和鼠标经过状态定义背景图像。

超链接的宽度可以不必等于背景图像的宽度，只要小于背景图的宽度即可。但是高度必须保持与背景图像的一致。在设计中可以结合背景图像的效果定义字体颜色。

第 4 步，在浏览器中预览，所得的超链接效果如图 6.15 所示。



图 6.15　背景图交换链接效果

提示： 为了减少两幅背景图像的 HTTP 请求次数，避免占用不必要的带宽，可以把用于交换的两幅图像合并为一幅图像，然后利用 CSS 定位技术控制背景图像的显示区域，详细代码请参考本节示例源码文件中的 test2.html。


视频讲解


Note

6.3.4 设计水平滑动菜单

CSS 滑动门的形式有两种：水平滑动和垂直滑动。6.3.2 节简单演示了图像滑动的基本方法，本节在此基础上介绍更复杂的案例。

【操作步骤】

第 1 步，设计“门”。这个门实际上就是背景图，滑动门一般至少需要两幅背景图，以实现闭合成门的设计效果，本例则完全采用一幅背景图像，一样能够设计出滑动门效果，如图 6.16 所示。考虑到门能够适应不同尺寸的菜单，所以背景图像的宽度和高度应该尽量大，这样就可以保证比较大的灵活性。

图 6.16 设计滑动门背景图

第 2 步，设计“门轴”。至少需要两个元素配合使用才能使门自由推拉。背景图需要安装在对应的门轴之上才能够自由推拉，从而呈现滑动效果。一般在列表结构中，可以将<li>和<a>标签配合使用。

第 3 步，启动 Dreamweaver，新建网页，保存为 test.html，在<body>标签内编写如下列表结构，由于每个菜单项字数不尽相同，使用滑动门来设计效果会更好。

```
<ul id="menu">
    <li><a href="#" title="">首页</a></li>
    <li><a href="#" title="">微博圈</a></li>
    <li><a href="#" title="">移动开发</a></li>
    <li><a href="#" title="">编程与设计</a></li>
    <li><a href="#" title="">程序员与语言</a></li>
    <li><a href="#" title="">编程语言排行榜</a></li>
</ul>
```

第 4 步，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后准备编写样式。

第 5 步，整理设计思路：

☑ 在下面叠放的标签（<li>）中定义如图 6.16 所示的背景图，并定位左对齐，使其左侧与<li>标签左侧对齐。

☑ 在上面叠放的标签（<a>）中设置相同的背景图，使其右侧与<a>标签的右侧对齐，这样两个背景图像就可以重叠在一起。

第 6 步，为了避免上下重叠元素的背景图相互挤压，导致菜单项两端的圆角背景图被覆盖，可以为<li>标签左侧和<a>标签右侧增加补白（padding），以此限制两个元素不能覆盖两端圆角背景图。

第 7 步，根据第 5 步和第 6 步的设计思路，动手编写如下 CSS 样式代码：

```
#menu {/*定义列表样式*/
    background: url(images/bg1.gif) #fff;        /*定义导航菜单的背景图像*/
    padding-left: 32px;                          /*定义左侧的补白*/
    margin: 0px;                                 /*清除边界*/
    list-style-type: none;                       /*清除项目符号*/
    height:35px;                                 /*固定高度，否则会自动收缩为0*/
}
#menu li {/*定义列表项样式*/
```



```
        float: left;                                          /*向左浮动*/
        margin:0 4px;                                         /*增加菜单项之间的距离*/
        padding-left:18px;                                    /*定义左补白，避免左侧圆角覆盖*/
        background:url(images/menu4.gif) left center repeat-x; /*定义背景图，左对齐*/
    }
    #menu li a {/*定义超链接默认样式*/
        padding-right: 18px;                                  /*定义右补白，与左侧形成对称*/
        float: left;                                          /*向左浮动*/
        height: 35px;                                         /*固定高度*/
        color: #bbb;                                          /*定义百分比宽度，实现与 li 同宽*/
        line-height: 35px;                                    /*定义行高，间接实现垂直对齐*/
        text-align: center;                                   /*定义文本水平居中*/
        text-decoration: none;                                /*清除下画线效果*/
        background:url(images/menu4.gif) right center repeat-x; /*定义背景图像*/
    }
    #menu li a:hover {/*定义鼠标经过超链接的样式*/
        text-decoration:underline;                            /*定义下画线*/
        color: #fff                                           /*白色字体*/
    }
```



第 8 步，保存页面，在浏览器中预览，演示效果如图 6.17 所示。

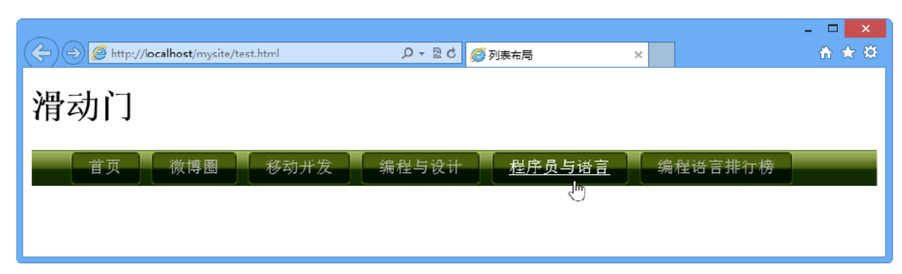


图 6.17　水平滑动菜单

6.3.5　设计垂直滑动菜单



6.3.3 节介绍了背景图像的垂直交换样式，但是单纯的垂直滑动存在一个弱点：如果菜单项字数不同（菜单项宽度不同），那么就需要考虑为不同宽度的菜单项设计背景图，这样就比较麻烦。解决方法为：将水平和垂直滑动融合在一起，设计菜单项能自由适应高度和宽度的变化。

【操作步骤】

第 1 步，设计背景图，如图 6.18 所示。然后将两幅背景图拼合在一起，形成滑动的门，如图 6.19 所示。



图 6.18　设计滑动背景图　　　　图 6.19　拼合滑动背景图

第 2 步，完善 HTML 结构，在超链接（<a>）内再包裹一层标签（<span>）。启动 Dreamweaver，新建网页，保存为 test.html，在<body>标签内编写如下列表结构。



Note

```
<h1>滑动门</h1>
<ul id="menu">
    <li><a href="#" title=""><span>首页</span></a></li>
    <li><a href="#" title=""><span>微博圈</span></a></li>
    <li><a href="#" title=""><span>移动开发</span></a></li>
    <li><a href="#" title=""><span>编程与设计</span></a></li>
    <li><a href="#" title=""><span>程序员与语言</span></a></li>
    <li><a href="#" title=""><span>编程语言排行榜</span></a></li>
</ul>
```

第 3 步，在<head>标签内添加<style type="text/css">标签，定义内部样式表，准备编写样式。

第 4 步，设计 CSS 样式代码，可参考 6.3.4 节示例样式代码，把<li>标签的背景样式转给<span>标签即可，详细代码如下：

```
#menu {/*定义列表样式*/
    background: url(images/bg1.gif) #fff;                    /*定义导航菜单的背景图像*/
    padding-left: 32px;                                      /*定义左侧的补白*/
    margin: 0px;                                             /*清除边界*/
    list-style-type: none;                                   /*清除项目符号*/
    height:35px;                                             /*固定高度，否则会自动收缩为 0*/
}
#menu li {/*定义列表项样式*/
    float: left;                                             /*向左浮动*/
    margin:0 4px;                                            /*增加菜单项之间的距离*/
}
#menu span {/*定义超链接内包含元素 span 的样式*/
     float:left;                                             /*向左浮动*/
    padding-left:18px;                                       /*定义左补白，避免左侧被覆盖*/
    background:url(images/menu4.gif) left center repeat-x;   /*定义背景图，左对齐*/
}
#menu li a {/*定义超链接默认样式*/
    padding-right: 18px;                                     /*定义右补白，与左侧形成对称*/
    float: left;                                             /*向左浮动*/
    height: 35px;                                            /*固定高度*/
    color: #bbb;                                             /*定义百分比宽度，实现与 li 同宽*/
    line-height: 35px;                                       /*定义行高，间接实现垂直对齐*/
    text-align: center;                                      /*定义文本水平居中*/
    text-decoration: none;                                   /*清除下画线效果*/
    background:url(images/menu4.gif) right center repeat-x;  /*定义背景图像*/
}
#menu li a:hover {/*定义鼠标经过超链接的样式*/
    text-decoration:underline;                               /*定义下画线*/
    color: #fff                                              /*白色字体*/
}
```

第 5 步，第 4 步样式代码仅完成了水平滑动效果，下面需要修改部分样式，设计当鼠标经过时的滑动效果，把如下样式：

```
#menu li a:hover {/*定义鼠标经过超链接的样式*/
    text-decoration:underline;                               /*定义下画线*/
```



```
    color: #fff                                          /*白色字体*/
}
```

修改为如下样式：

```
#menu a:hover {/*定义鼠标经过超链接的样式*/
    color: #fff;                                         /*白色字体*/
    background:url(images/menu5.gif) right center repeat-x;  /*定义滑动后的背景图像*/
}
#menu a:hover span {/*定义鼠标经过超链接的样式*/
    background:url(images/menu5.gif) left center repeat-x;   /*定义滑动后的背景图像*/
    cursor:pointer;                                      /*定义鼠标经过时显示手形指针*/
    cursor:hand;                                         /*早期 IE 版本下显示为手形指针*/
}
```



第 6 步，保存页面之后，在浏览器中预览，则演示效果如图 6.20 所示。



图 6.20　水平与垂直滑动菜单

提示： 如果使用 CSS3 动画技术，添加如下两个样式，可以更逼真地演示垂直滑动的动画效果（test3.html），相关技术的详细讲解可以参考后面章节内容。

```
#menu span {transition: all .3s ease-in;}
#menu li a {transition: all .3s ease-in;}
```

6.3.6　设计 Tab 选项面板



视 频 讲 解

Tab 在栏目面板中比较常用，因为它能够在有限的空间内包含更多分类信息，适合商业网站的版面集成设计。

设计思路：利用 CSS 隐藏或显示栏目的部分内容，实际 Tab 面板所包含的全部内容都已经下载到客户端浏览器中。一般 Tab 面板仅显示一个 Tab 菜单项，当用户选择对应的菜单项之后，才会显示对应的内容。

【操作步骤】

第 1 步，启动 Dreamweaver，新建网页，保存为 test.html，在<body>标签内编写如下结构，构建 HTML 文档。

```
<div id="tab">
    <div class="Menubox">
        <ul>
            <li id="tab_1" class="hover" onclick="setTab(1,4)">明星</li>
```



Note

```
                <li id="tab_2" onclick="setTab(2,4)">搞笑</li>
                <li id="tab_3" onclick="setTab(3,4)">美女</li>
                <li id="tab_4" onclick="setTab(4,4)">摄影</li>
            </ul>
        </div>
        <div class="Contentbox">
            <div id="con_1" class="hover" ><img src="images/1.png" /></div>
            <div id="con_2" class="hide"><img src="images/2.png" /></div>
            <div id="con_3" class="hide"><img src="images/3.png" /></div>
            <div id="con_4" class="hide"><img src="images/4.png" /></div>
        </div>
    </div>
```

在 Tab 面板中，<div class="Menubox">框包含的内容是菜单栏，<div class="Contentbox">框包含的是面板内容。

第 2 步，在<head>标签内添加<style type="text/css">标签，定义内部样式表，准备编写样式。

第 3 步，定义 Tab 菜单的 CSS 样式。这里包含 3 部分 CSS 代码：第 1 部分重置列表框、列表项和超链接默认样式，第 2 部分定义 Tab 选项卡基本结构，第 3 部分定义与 Tab 菜单相关的几个类样式。详细代码如下：

```
/*页面元素的默认样式*/
a {/*超链接的默认样式*/
    color:#00F;                          /*定义超链接的默认颜色*/
    text-decoration:none;                /*清除超链接的下画线样式*/
}
a:hover {color: #c00;}/*鼠标经过超链接的默认样式*/
ul {/*定义列表结构基本样式*/
    list-style:none;                     /*清除默认的项目符号*/
    padding:0;                           /*清除补白*/
    margin:0px;                          /*清除边界*/
    text-align:center;                   /*定义包含文本居中显示*/
}
/*选项卡结构*/
#tab {/*定义选项卡的包含框样式*/
    width:920px;                         /*定义 Tab 面板的宽度*/
    margin:0 auto;                       /*定义 Tab 面板居中显示*/
    font-size:12px;                      /*定义 Tab 面板的字体大小*/
    overflow:hidden;                     /*隐藏超出区域的内容*/
}
/*菜单样式类*/
.Menubox {/*Tab 菜单栏的类样式*/
    width:100%;                          /*定义宽度，满包含框宽度显示*/
    background:url(images/tab1.gif);     /*定义 Tab 菜单栏的背景图像*/
    height:28px;                         /*固定高度*/
    line-height:28px;                    /*定义行高，实现垂直居中显示*/
}
.Menubox ul {margin:0px; padding:0px;}/*清除列表缩进样式*/
.Menubox li {/*Tab 菜单栏包含的列表项基本样式*/
```





```
        float:left;                              /*向左浮动，实现并列显示*/
        display:block;                           /*块状显示*/
        cursor:pointer;                          /*定义手形指针样式*/
        width:114px;                             /*固定宽度*/
        text-align:center;                       /*定义文本居中显示*/
        color:#949694;                           /*字体颜色*/
        font-weight:bold;                        /*加粗字体*/
    }
    .Menubox li img{width:100%;}
    .Menubox li.hover {/*鼠标经过列表项的样式类*/
        padding:0px;                             /*清除补白*/
        background:#fff;                         /*加亮背景色*/
        width:116px;                             /*固定宽度显示*/
        border:1px solid #A8C29F;                /*定义边框线*/
        border-bottom:none;                      /*清除底边框线样式*/
        background:url(images/tab2.gif);         /*定义背景图像*/
        color:#739242;                           /*定义字体颜色*/
        height:27px;                             /*固定高度*/
        line-height:27px;                        /*定义行高，实现文本垂直居中*/
    }
    .Contentbox {/*定义 Tab 面板中内容包含框基本样式类*/
        clear:both;                              /*清除左右浮动元素*/
        margin-top:0px;                          /*清除顶边界*/
        border:1px solid #A8C29F;                /*定义边框线样式*/
        border-top:none;                         /*清除顶部边框线样式*/
        padding-top:8px;                         /*定义顶部补白，增加距离*/
    }
    .hide {display:none; /*隐藏元素显示*/}         /*隐藏样式类*/
```

第 4 步，使用 JavaScript 设计 Tab 交互效果。

下面函数定义了两个参数，第一个参数定义要隐藏或显示的面板，第二个参数定义当前 Tab 面板包含几个 Tab 选项卡，并定义当前选项卡包含的列表项的类样式为 hover，最后为每个 Tab 菜单中的 li 元素调用该函数即可，从而实现单击对应的菜单项，即可自动激活该脚本函数，并把当前列表项的类样式设置为 hover，同时显示该菜单对应的面板内容，而隐藏其他面板内容。有关 JavaScript 语言的详细讲解可以参考后面章节内容。

```
<script>
function setTab(cursel,n){
    for(i=1;i<=n;i++){
        var menu=document.getElementById("tab_"+i);
        var con=document.getElementById("con_"+i);
        menu.className=i==cursel?"hover":"";
        con.style.display=i==cursel?"block":"none";
    }
}
</script>
```

第 5 步，保存页面之后，在浏览器中预览，则演示效果如图 6.21 所示。



Note

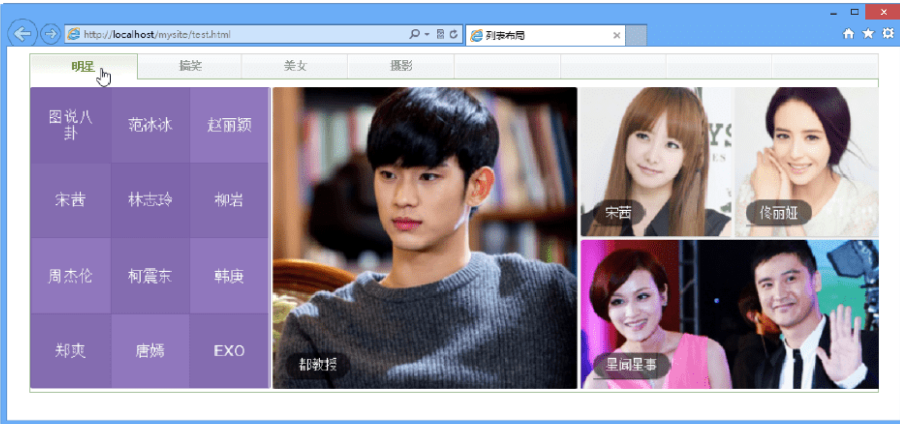


图 6.21　Tab 面板菜单效果



视频讲解

6.3.7　设计下拉式面板

下拉式面板比较特殊，当鼠标移到菜单项目上时将自动弹出一个下拉的大面板，在该面板中显示各种分类信息。这种版式在电商类型网站中应用比较多。

设计思路：在超链接（<a>标签）内包含面板结构，当鼠标移过超链接时，自动显示这个面板，而在默认状态隐藏其显示。由于早期 IE 浏览器对<a>标签包含其他结构的解析存在问题，设计时应适当考虑兼容问题。

【操作步骤】

第 1 步，启动 Dreamweaver，新建网页，保存为 test.html，在<body>标签内编写如下结构，构建 HTML 文档。

```
<ul id="lists">
    <li><a href="#" class="tl">商品导购
        <!--[if IE 7]><!--></a><!--<![endif]-->
        <!--[if lte IE 6]><table><tr><td><![endif]-->
        <div class="pos1">
            <dl id="menu">
                <dt>产品大类</dt>
                    <dd><a href="#" title="">图书、音像、数字商品</a></dd>
                    <dd><a href="#" title="">家用电器</a></dd>
                    <dd><a href="#" title="">手机、数码、京东通信预约</a></dd>
                    <dd><a href="#" title="">电脑、办公</a></dd>
                    <dd><a href="#" title="">家居、家具、家装、厨具</a></dd>
                    <dd><a href="#" title="">服饰内衣、珠宝首饰</a></dd>
                    <dd><a href="#" title="">个护化妆</a></dd>
                    <dd><a href="#" title="">鞋靴、箱包、钟表、奢侈品</a></dd>
                    <dd><a href="#" title="">运动户外</a></dd>
            </dl>
        </div>
        <!--[if lte IE 6]></td></tr></table></a><![endif]-->
    </li>
</ul>
```



提示：在超链接中包含一个面板结构，为了让超链接在 IE 浏览器中能够正常响应，代码中使用 IE 条件语句（后面章节会详细讲解）。IE 条件语句是一个条件结构，用来判断当前 IE 浏览器的版本号，以便执行不同的 CSS 样式或解析不同的 HTML 结构。



Note

第 2 步，在<head>标签内添加<style type="text/css">标签，定义内部样式表，准备编写样式。

第 3 步，编写下拉式导航面板的 CSS 样式如下：

```
#lists {/*定义总包含框基本结构*/
    background: url(images/bg1.gif) #fff;                    /*背景图像*/
    padding-left: 32px;                                       /*左侧补白*/
    margin: 0px;                                              /*清除边界*/
    height:35px;                                              /*固定高度*/
    font-size:12px;                                           /*字体大小*/
}
#lists li {/*定义列表项目基本样式*/
    display:inline;                                           /*行内显示*/
    float:left;                                               /*向左浮动*/
    height:35px;                                              /*固定高度*/
    background:url(images/menu5.gif) no-repeat left center;   /*背景图像*/
    padding-left:12px;                                        /*左侧补白*/
    position:relative;                        /*相对定位，为下拉导航面板绝对定位指定一个参考框*/
}
#lists li a.tl {/*定义超链接基本样式*/
    display:block;                                            /*块状显示*/
    width:80px;                                               /*固定宽度*/
    height:35px;                                              /*固定高度*/
    text-decoration:none;                                     /*清除下画线*/
    text-align:center;                                        /*文本水平居中*/
    line-height:35px;                                         /*行高，实现垂直居中*/
    font-weight:bold;                                         /*加粗显示*/
    color:#fff;                                               /*字体颜色为白色*/
    background:url(images/menu5.gif) no-repeat right center;  /*定义导航背景图像*/
    padding-right:12px;                                       /*定义右侧补白大小*/
}
#lists div {display:none;}/*定义超链接包含的导航面板的隐藏显示*/
#lists :hover div {/*显示并定义超链接包含的导航面板*/
    display:block;                                            /*块状显示*/
    width:598px;                                              /*固定宽度*/
    background:#faebd7;                                       /*定义背景色*/
    position:absolute;                                        /*绝对定位，以便自由显示*/
    left:1px;                                                 /*距离包含框左侧（li 元素）的距离*/
    top:34px;                                                 /*距离包含框顶部的距离*/
    border:1px solid #888;                                    /*定义边框线*/
    padding-bottom:10px;                                      /*定义底部补白*/
}
```

第 4 步，保存页面之后，在浏览器中预览，则演示效果如图 6.22 所示。





Note



图 6.22　下拉式导航面板

6.4　在 线 练 习

本节通过大量案例练习 CSS3 列表和超链接的样式设计，感兴趣的读者可以扫码强化基本功训练：（1）列表样式；（2）超链接样式。



在线练习 1



在线练习 2



使用 CSS 美化表格

在传统网页设计中，表格的主要功能就是页面布局，因此表格也成为网页编辑的工具；在标准化网页设计中，表格的主要功能是显示数据，也可适当辅助结构设计。本章将详细介绍表格在网页设计中的应用，包括设计符合标准的表格结构、正确设置表格属性和灵活应用 CSS 表格样式。

【学习重点】

- 正确使用表格标签。
- 设置表格和单元格属性。
- 设计表格的 CSS 样式。



7.1 表格属性





视频讲解

本节将介绍<table>、<td>和<th>的标签属性。

7.1.1 设置表格属性

表格标签<table>包含大量属性，其中大部分属性都可以使用 CSS 属性代替，也有几个专用属性无法使用 CSS 实现。HTML5 支持的<table>标签属性说明如表 7.1 所示。

表 7.1 HTML5 支持的<table>标签属性

属 性	说 明
border	定义表格边框，值为整数，单位为像素。当值为 0 时，表示隐藏表格边框线。功能类似 CSS 中的 border 属性，但是没有 CSS 提供的边框属性强大
cellpadding	定义数据表单元格的补白。功能类似 CSS 中的 padding 属性，但是功能比较弱
cellspacing	定义数据表单元格的边界。功能类似 CSS 中的 margin 属性，但是功能比较弱
width	定义数据表的宽度。功能类似 CSS 中的 width 属性
frame	设置数据表的外边框线显示，实际上它是对 border 属性的功能扩展，取值包括 void（不显示任一边框线）、above（顶端边框线）、below（底部边框线）、hsides（顶部和底部边框线）、lhs（左边框线）、rhs（右边框线）、vsides（左和右边的框线）、box（所有四周的边框线）、border（所有四周的边框线）
rules	设置数据表的内边线显示，实际上它是对 border 属性的功能扩展，取值包括 none（禁止显示内边线）、groups（仅显示分组内边线）、rows（显示每行的水平线）、cols（显示每列的垂直线）、all（显示所有行和列的内边线）
summary	定义表格的摘要，没有 CSS 对应属性

rules 和 frame 是两个特殊的表格样式属性，用于定义表格的各个内、外边框线是否显示。由于使用 CSS 的 border 属性可以实现相同的效果，所以不建议用户选用。

【示例 1】 下面示例借助表格标签的 frame 和 rules 属性定义表格以单行线的形式进行显示。

```
<table border="1" frame="hsides"  rules="rows" width="100%">
    <caption>frame 属性取值说明</caption>
    <tr><th>值</th><th>说明</th></tr>
    <tr><td>void</td><td>不显示外侧边框。</td></tr>
    <tr><td>above</td><td>显示上部的外侧边框。</td></tr>
    <tr><td>below</td><td>显示下部的外侧边框。</td> </tr>
    <tr><td>hsides</td><td>显示上部和下部的外侧边框。</td></tr>
    <tr><td>vsides</td><td>显示左边和右边的外侧边框。</td></tr>
    <tr><td>lhs</td><td>显示左边的外侧边框。</td></tr>
    <tr><td>rhs</td><td>显示右边的外侧边框。</td></tr>
    <tr><td>box</td> <td>在所有四个边上显示外侧边框。</td></tr>
    <tr><td>border</td><td>在所有四个边上显示外侧边框。</td></tr>
</table>
```



上面示例通过 frame 属性定义表格仅显示上下框线，使用 rules 属性定义表格仅显示水平内边线，从而设计出单行线数据表格效果。在使用 frame 和 rules 属性时，同时定义 border 属性，指定数据表显示边框线。在浏览器中预览，则显示效果如图 7.1 所示。



cellpadding 属性用于定义单元格边沿与其内容之间的空白，cellspacing 属性定义单元格之间的空间。这两个属性的取值单位为像素或者百分比。

【示例 2】下面示例设计井字形状的表格。

```
<table border="1" frame="void" cellpadding="6" cellspacing="16">
    <caption>rules 属性取值说明</caption>
    <tr><th>值</th><th>说明</th></tr>
    <tr><td>none</td><td>没有线条。</td></tr>
    <tr><td>groups</td><td>位于行组和列组之间的线条。</td></tr>
    <tr><td>rows</td><td>位于行之间的线条。</td></tr>
    <tr><td>cols</td><td>位于列之间的线条。</td></tr>
    <tr><td>all</td><td>位于行和列之间的线条。</td></tr>
</table>
```

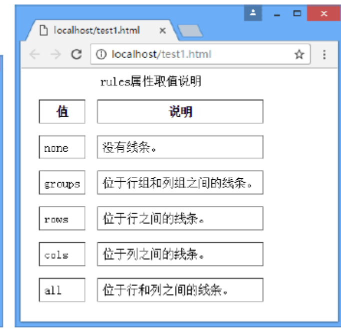
上面示例通过 frame 属性隐藏表格外框，然后使用 cellpadding 属性定义单元格内容的边距为 6px，单元格之间的间距为 16px，则在浏览器中预览效果如图 7.2 所示。



frame属性取值说明

值	说明
void	不显示外侧边框。
above	显示上部的外侧边框。
below	显示下部的外侧边框。
hsides	显示上部和下部的外侧边框。
vsides	显示左边和右边的外侧边框。
lhs	显示左边的外侧边框。
rhs	显示右边的外侧边框。
box	在所有四个边上显示外侧边框。
border	在所有四个边上显示外侧边框。

图 7.1　定义单线表格样式



rules属性取值说明

值	说明
none	没有线条。
groups	位于行组和列组之间的线条。
rows	位于行之间的线条。
cols	位于列之间的线条。
all	位于行和列之间的线条。

图 7.2　定义井字表格样式

提示：cellpadding 属性定义的效果可以使用 CSS 的 padding 样式属性代替，建议不要使用 cellpadding 属性。

7.1.2　设置单元格属性



单元格标签（<td>和<th>）包含大量属性，其中大部分属性都可以使用 CSS 属性代替，也有几个专用属性无法使用 CSS 实现。HTML5 支持的<td>和<th>标签属性说明如表 7.2 所示。

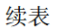
表 7.2　HTML5 支持的<td>和<th>标签属性

属　性	说　明
abbr	定义单元格中内容的缩写版本
align	定义单元格内容的水平对齐方式，取值包括 right（右对齐）、left（左对齐）、center（居中对齐）、justify（两端对齐）和 char（对准指定字符）。功能类似 CSS 中的 text-align 属性，建议使用 CSS 完成设计





Note



续表

属　　性	说　　明
axis	对单元格进行分类，取值为一个类名
char	定义根据哪个字符来进行内容的对齐
charoff	定义对齐字符的偏移量
colspan	定义单元格可横跨的列数
headers	定义与单元格相关的表头
rowspan	定义单元格可横跨的行数
scope	定义将表头数据与单元格数据相关联的方法，取值包括 col（列的表头）、colgroup（列组的表头）、row（行的表头）、rowgroup（行组的表头）
valign	定义单元格内容的垂直排列方式。取值包括 top（顶部对齐）、middle（居中对齐）、bottom（底部对齐）、baseline（基线对齐）。功能类似 CSS 中的 vertical-align 属性，建议使用 CSS 完成设计

colspan 和 rowspan 是两个重要的单元格属性，分别用来定义单元格可跨列或跨行显示。取值为正整数，如果取值为 0，则表示浏览器横跨到列组的最后一列或者行组的最后一行。

【示例】下面示例使用 colspan=5 属性，定义单元格跨列显示，效果如图 7.3 所示。

```
<table border=1>
    <tr><th align=center colspan=5>课程表</th></tr>
    <tr><th>星期一</th><th>星期二</th> <th>星期三</th><th>星期四</th><th>星期五</th></tr>
    <tr><td align=center colspan=5>上午</td> </tr>
    <tr><td>语文</td><td>物理</td> <td>数学</td> <td>语文</td><td>美术</td></tr>
    <tr><td>数学</td><td>语文</td><td>体育</td> <td>英语</td><td>音乐</td></tr>
    <tr><td>语文</td> <td>体育</td><td>数学</td><td>英语</td><td>地理</td></tr>
    <tr><td>地理</td><td>化学</td><td>语文</td> <td>语文</td><td>美术</td></tr>
    <tr><td align=center colspan=5>下午</td></tr>
    <tr><td>作文</td><td>语文</td><td>数学</td><td>体育</td><td>化学</td></tr>
    <tr> <td>生物</td><td>语文</td><td>物理</td><td>自修</td><td>自修</td></tr>
</table>
```

图 7.3　定义单元格跨列显示

7.2　表格基本样式

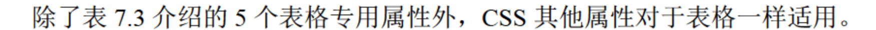
CSS 为表格定义了 5 个专用属性，详细说明如表 7.3 所示。





表 7.3　CSS 表格属性列表

属　　性	取　　值	说　　明
border-collapse	separate（边分开）｜collapse（边合并）	定义表格的行和单元格的边是合并在一起还是按照标准的 HTML 样式分开
border-spacing	length	定义当表格边框独立（如 border-collapse 属性等于 separate）时，行和单元格的边在横向和纵向上的间距，该值不可以取负值
caption-side	top | bottom	定义表格的 caption 对象位于表格的顶部或底部。应与 caption 元素一起使用
empty-cells	show | hide	定义当单元格无内容时，是否显示该单元格的边框
table-layout	auto | fixed	定义表格的布局算法，可以通过该属性改善表格呈递性能，如果设置 fixed 属性值，会使 IE 以一次一行的方式呈递表格内容，从而以更快的速度将信息提供给用户；如果设置 auto 属性值，则表格在每一单元格内所有内容读取计算之后才会显示出来



除了表 7.3 介绍的 5 个表格专用属性外，CSS 其他属性对于表格一样适用。

7.2.1　设计表格边框线



使用 CSS 的 border 属性代替<table>标签的 border 属性定义表格边框，可以优化代码结构。

【示例】下面示例演示如何使用 CSS 设计细线边框样式的表格。

【操作步骤】

第 1 步，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表。

第 2 步，在内部样式表中输入下面样式代码，定义单元格边框显示为 1px 的灰色实线。

```
th, td {font-size:12px; border:solid 1px gray;}
```

第 3 步，在<body>标签内构建一个简单的表格结构。

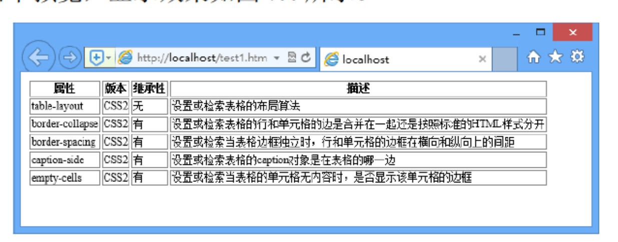
```
<table>
    <tr>
        <th>属性</th>
        <th>版本</th>
        <th>继承性</th>
        <th>描述</th>
    </tr>
    <tr>
        <td>table-layout</td>
        <td>CSS2</td>
        <td>无</td>
        <td>设置或检索表格的布局算法</td>
    </tr>
    <tr>
        <td>border-collapse</td>
        <td>CSS2</td>
        <td>有</td>
        <td>设置或检索表格的行和单元格的边是合并在一起还是按照标准的 HTML 样式分开</td>
    </tr>
```






```
    <tr>
        <td>border-spacing</td>
        <td>CSS2</td>
        <td>有</td>
        <td>设置或检索当表格边框独立时，行和单元格的边框在横向和纵向上的间距</td>
    </tr>
    <tr>
        <td>caption-side</td>
        <td>CSS2</td>
        <td>有</td>
        <td>设置或检索表格的 caption 对象是在表格的哪一边</td>
    </tr>
    <tr>
        <td>empty-cells</td>
        <td>CSS2</td>
        <td>有</td>
        <td>设置或检索当表格的单元格无内容时，是否显示该单元格的边框</td>
    </tr>
</table>
```

第 4 步，在浏览器中预览，显示效果如图 7.4 所示。



属性	版本	继承性	描述
table-layout	CSS2	无	设置或检索表格的布局算法
border-collapse	CSS2	有	设置或检索表格的行和单元格的边是合并在一起还是按照标准的HTML样式分开
border-spacing	CSS2	有	设置或检索当表格边框独立时，行和单元格的边框在横向和纵向上的间距
caption-side	CSS2	有	设置或检索表格的caption对象是在表格的哪一边
empty-cells	CSS2	有	设置或检索当表格的单元格无内容时，是否显示该单元格的边框

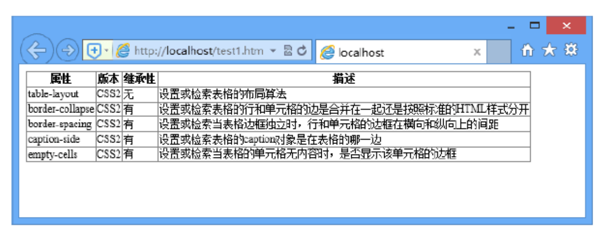
图 7.4　使用 CSS 定义单元格边框样式

通过效果图可以看到，使用 CSS 定义的单行线不是连贯的线条。这是因为表格中每个单元格都是一个独立的空间，为它们定义边框线时，相互之间不是紧密连接在一起的。

第 5 步，在内部样式表中，为 table 元素添加如下 CSS 样式，把相邻单元格进行合并。

```
table {border-collapse:collapse;}/*合并单元格边框*/
```

第 6 步，在浏览器中重新预览，则页面效果如图 7.5 所示。



属性	版本	继承性	描述
table-layout	CSS2	无	设置或检索表格的布局算法
border-collapse	CSS2	有	设置或检索表格的行和单元格的边是合并在一起还是按照标准的HTML样式分开
border-spacing	CSS2	有	设置或检索当表格边框独立时，行和单元格的边框在横向和纵向上的间距
caption-side	CSS2	有	设置或检索表格的caption对象是在表格的哪一边
empty-cells	CSS2	有	设置或检索当表格的单元格无内容时，是否显示该单元格的边框

图 7.5　使用 CSS 合并单元格边框



视频讲解

7.2.2　定义单元格间距和空隙

为了兼容<table>标签的 cellspacing 属性，CSS 定义了 border-spacing 属性，该属性能够分离单元格



间距，取值包含一个或两个值。当定义一个值时，则定义单元格行间距和列间距都为该值。例如：

```
table {border-spacing:20px;}/*分隔单元格边框*/
```

如果分别定义行间距和列间距，就需要定义两个值，例如：

```
table {border-spacing:10px 30px;}/*分隔单元格边框*/
```



其中第一个值表示单元格之间的行间距，第二个值表示单元格之间的列间距，该属性值不可以为负数。使用 cellspacing 属性定义单元格之间的距离之后，该空间由表格背景填充。

使用该属性应注意几个小问题：

☑ 早期 IE 浏览器不支持该属性，要定义相同效果的样式，这就需要结合传统<table>标签的 cellspacing 属性来设置。

☑ 使用 cellspacing 属性时，应确保单元格之间相互独立性，不能使用 border-collapse:collapse;样式定义合并表格内单元格的边框。

☑ cellspacing 属性不能够使用 CSS 的 margin 属性来代替。对于 td 元素来说，不支持 margin 属性。

☑ 可以为单元格定义补白，此时使用 CSS 的 padding 属性与单元格的 cellpadding 标签属性实现效果是相同的。

【示例 1】以 7.2.1 节示例中表格结构为基础，重新设计内部样式表，为表格内单元格定义上下 6px 和左右 12px 的间距，同时设计单元格内部空隙为 12px，演示效果如图 7.6 所示。

```
table {border-spacing: 6px 12px;}
th, td {
    font-size: 12px;
    border: solid 1px gray;
    padding: 12px;
}
```



属性	版本	继承性	描述
table-layout	CSS2	无	设置或检索表格的布局算法
border-collapse	CSS2	有	设置或检索表格的行和单元格的边是合并在一起还是按照标准的HTML样式分开
border-spacing	CSS2	有	设置或检索当表格边框独立时，行和单元格的边框在横向和纵向上的间距
caption-side	CSS2	有	设置或检索表格的caption对象是在表格的哪一边
empty-cells	CSS2	有	设置或检索当表格的单元格无内容时，是否显示该单元格的边框

图 7.6　增加单元格空隙

也可以为<table>标签定义补白，此时可以增加表格外框与单元格之间的距离。

【示例 2】以示例 1 为基础，为<table>标签重设如下样式，设计表格外框为 2px 红色实线，定义表格外框与内部单元格间距为 2px，则显示效果如图 7.7 所示。

```
table {
    border-spacing: 6px 12px;
    border: solid 2px red;
```



```
    padding: 2px;
}
```





属性	版本	继承性	描述
table-layout	CSS2	无	设置或检索表格的布局算法
border-collapse	CSS2	有	设置或检索表格的行和单元格的边是合并在一起还是按照标准的HTML样式分开
border-spacing	CSS2	有	设置或检索当表格边框独立时，行和单元格的边框在横向和纵向上的间距
caption-side	CSS2	有	设置或检索表格的caption对象是在表格的哪一边
empty-cells	CSS2	有	设置或检索当表格的单元格无内容时，是否显示该单元格的边框

图 7.7　为表格和单元格同时定义补白效果



视频讲解

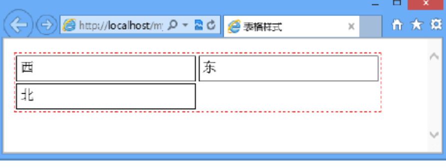
7.2.3　隐藏空单元格

如果表格单元格的边框处于分离状态（border-collapse:separate;），可以使用 CSS 的 empty-cells 属性设置空单元格是否显示。当其值为 show 时，表示显示空单元格；当其值为 hide 时，表示隐藏空单元格。

【示例】在下面示例中，隐藏第 2 行第 2 列的空单元格边框，效果如图 7.8 所示。

```
<style type="text/css">
table {/*表格样式*/
    width: 400px;                    /*固定表格宽度*/
    border: dashed 1px red;          /*定义虚线表格边框*/
    empty-cells: hide;               /*隐藏空单元格*/
}
th, td {/*单元格样式*/
    border: solid 1px #000;          /*定义实线单元格边框*/
    padding: 4px;                    /*定义单元格内的补白区域*/
}
</style>

<table>
    <tr><td>西</td><td>东</td> </tr>
    <tr><td>北</td><td></td></tr>
</table>
```



隐藏空白单元格　　　　默认显示的空白单元格

图 7.8　隐藏空单元格效果



> 提示：所谓空单元格，就是没有可视内容的单元格。如果单元格的 visibility 属性值为 hidden，即便单元格包含内容，也认为是无可视内容。而“ ”和其他空白字符为可视内容。ASCII 字符中的回车符（\0D）、换行符（\0A）、Tab 键（\09）和空格键（\20）表示无可视内容。如果表格行中所有单元格的 empty-cells 属性都为 hide，且都不包含任何可视内容，那么整行就等于设置了 display: none。



Note



视频讲解

7.2.4　定义标题样式

使用 CSS 的 caption-side 属性可以定义标题的显示位置，该属性取值包括 top（位于表格上面）、bottom（位于表格底部）、left（位于表格左侧，非标准）和 right（位于表格右侧，非标准）。

如果要水平对齐标题文本，则可以使用 text-align 属性。对于左右两侧的标题，可以使用 vertical-align 属性进行垂直对齐，取值包括 top、middle 和 bottom，其他取值无效，默认为 top。

【示例】在下面示例中，定义标题靠左显示，并设置标题垂直居中显示。但不同浏览器在解析时分歧比较大，如在 IE 浏览器中显示效果如图 7.9 所示，但是在 Firefox 中显示效果如图 7.10 所示。

```
<style type="text/css">
table {border: dashed 1px red; }                    /*定义表格虚线外框样式*/
th, td {/*定义单元格样式*/
    border: solid 1px #000;                         /*实线内框*/
    padding: 20px 80px;                             /*单元格内补白大小*/
}
caption {/*定义标题行样式*/
    caption-side: left;                             /*左侧显示*/
    width: 10px;                                    /*定义宽度*/
    margin: auto 20px;                              /*定义左右边界*/
    vertical-align: middle;                         /*垂直居中显示*/
    font-size: 14px;                                /*定义字体大小*/
    font-weight: bold;                              /*加粗显示*/
    color: #666;                                    /*灰色字体*/
}
</style>

<table>
    <caption>表格标题</caption>
    <tr><td>北</td><td>西</td> </tr>
    <tr><td>东</td><td>南</td> </tr>
</table>
```

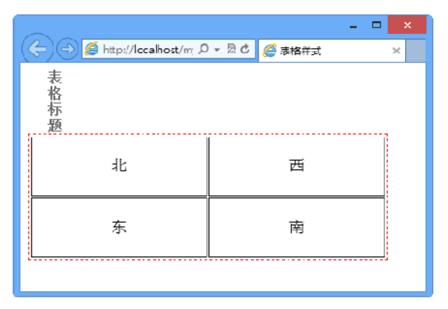


图 7.9　IE 解析表格标题效果

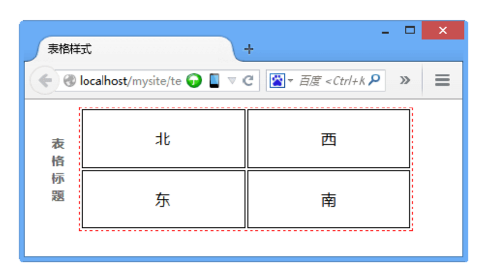


图 7.10　Firefox 解析表格标题效果




Note

【拓展】

当同时为<table>、<tr>和<td>等标签定义背景色、边框、字体属性等样式时，就容易发生样式重叠问题。根据表格布局模型，各种表格对象背景样式层叠的顺序如图 7.11 所示。

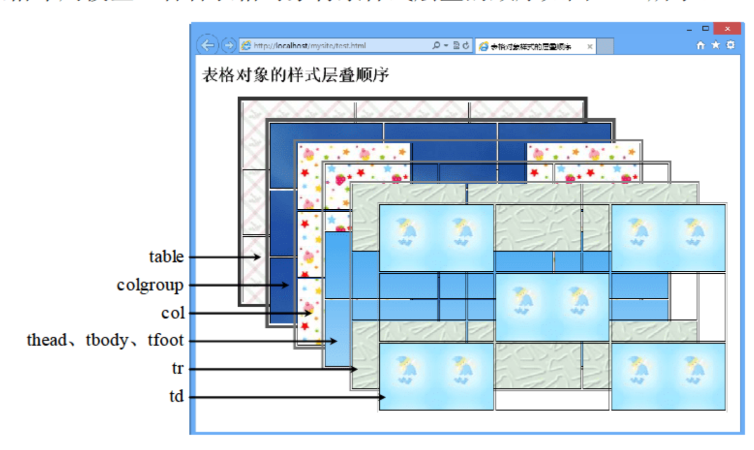


图 7.11　表格对象样式的层叠顺序

从图 7.11 可以看到：td 元素的样式具有最大优先权，以此类推，如果单元格为透明，则行（tr 元素）具有最大优先权。表格定义的背景优先权最小，当表格中其他元素都为透明时，才可以看到表格的背景。

7.3　案例实战

本节将结合几个案例详细讲解表格样式的一般设计技巧。


视频讲解

7.3.1　设计斑马线表格

本例将对一个简单的表格进行设计，使它看起来更为精致。另外，当表格的行和列都很多，并且数据量很大时，单元格采用相同的背景色会使浏览者感到凌乱，发生看错行的情况，此时可为表格设置隔行变色的效果，使得奇数行和偶数行的背景色不一样，示例效果如图 7.12 所示。

历届奥运会中国奖牌数

编号	年份	城市	金牌	银牌	铜牌	总计
第23届	1984年	洛杉矶（美国）	15	8	9	32
第24届	1988年	汉城（韩国）	5	11	12	28
第25届	1992年	巴塞罗那（西班牙）	16	22	16	54
第26届	1996年	亚特兰大（美国）	16	22	12	50
第27届	2000年	悉尼（澳大利亚）	28	16	15	59
第28届	2004年	雅典（希腊）	32	17	14	63
第29届	2008年	北京（中国）	51	21	28	100
第30届	2012年	伦敦（英国）	38	27	23	88
第31届	2016年	里约热内卢（巴西）	26	18	26	70
合计		544枚				

图 7.12　设计隔行变色的样式效果



【操作步骤】

第 1 步，新建 HTML5 文档，设计表格结构。

```
<table summary="历届奥运会中国奖牌数">
    <caption>历届奥运会中国奖牌数</caption>
        <tr><th>编号</th><th>年份</th><th>城市</th><th>金牌</th><th>银牌</th><th>铜牌</th><th>总计</th></tr>
    </thead>
    <tbody>
        <tr><td>第 23 届</td><td>1984 年</td><td>洛杉矶（美国）</td><td>15</td><td>8</td><td>9</td><td>32</td></tr>
        <tr><td>第 24 届</td><td>1988 年</td><td>汉城（韩国）</td><td> 5</td><td>11</td><td>12</td><td>28</td></tr>
        <tr><td>第 25 届</td><td>1992 年</td><td>巴塞罗那（西班牙）</td><td>16</td><td>22</td><td>16</td><td>54</td></tr>
        <tr><td>第 26 届</td><td>1996 年</td><td>亚特兰大（美国）</td><td>16</td><td>22</td><td>12</td><td>50</td></tr>
        <tr><td>第 27 届</td><td>2000 年</td><td>悉尼（澳大利亚）</td><td>28</td><td>16</td><td>15</td><td>59</td></tr>
        <tr><td>第 28 届</td><td>2004 年</td><td>雅典（希腊）</td><td>32</td><td>17</td><td>14</td><td>63</td></tr>
        <tr><td>第 29 届</td><td>2008 年</td><td>北京（中国）</td><td>51</td><td>21</td><td>28</td><td>100</td></tr>
        <tr><td>第 30 届</td><td>2012 年</td><td>伦敦（英国）</td><td>38</td><td>27</td><td>23</td><td>88</td></tr>
        <tr><td>第 31 届</td><td>2016 年</td><td>里约热内卢（巴西）</td><td>26</td><td>18</td><td>26</td><td>70</td></tr>
    </tbody>
    <tfoot>
        <tr><th>合计</th><td colspan="4">544 枚</td></tr>
    </tfoot>
</table>
```

在这个表格中，使用的标记从上至下依次为<caption>、<thead>、<tbody>和<tfoot>，分别定义表格的标题、列标题行、数据区域和脚注行。

第 2 步，在头部区域<head>标签中插入一个<style type="text/css">标签，在该标签中输入下面样式代码，定义表格样式和表格标题样式。

```
table {/*表格样式*/
    background-color: #FFF; color: #565;    /*表格背景色和字体颜色*/
    border: none;                           /*清除表格外框线*/
    font: 12px arial;                       /*设置字体大小和类型*/
    width: 90%;                             /*定义弹性宽度*/
    margin: 12px auto;                      /*表格水平居中显示*/
}
table caption {/*表格标题样式*/
    font-size: 24px;                        /*定义表格标题字体大小*/
    border-bottom: 2px solid #B3DE94;       /*添加下边线*/
    border-top: 2px solid #B3DE94;          /*添加上边线*/
}
```

第 3 步，设计数据行单元格样式。



Note

```
tbody td, tbody th {
    background-color: #DFC;                /*定义背景色*/
    border-bottom: 2px solid #B3DE94;      /*添加下边线*/
    border-top: 3px solid #FFFFFF;         /*添加上边线*/
    padding: 9px;                          /*增大单元格空间*/
}
```

第 4 步，设计脚注行单元格样式。

```
tfoot td, tfoot th {
    font-weight: bold;                     /*加粗显示*/
    padding: 4px 8px 6px 9px;              /*增大单元格空间*/
    text-align: center;                    /*居中显示*/
}
```

第 5 步，设计列标题行样式。

```
thead th {
    font-size: 14px; line-height: 19px;
    padding: 0 8px 2px;
}
```

第 6 步，为最后 4 列单元格定义样式。

```
tbody td+td+td+td{
    color:red;
    text-align:right;
}
```

这里使用 3 个相邻选择器组合使用，精确匹配到后面 4 列单元格标签，然后设置字体颜色为红色，文本右对齐显示。

【拓展】

用户可以使用<col>或<colgroup>标签为各列定义样式。例如，下面代码中第 2 个<col span="4">标签可以为最后 4 列定义样式。

```
<table summary="历届奥运会中国奖牌数">
    <caption>
    历届奥运会中国奖牌数
    </caption>
    <col span="3"></col>
    <col span="4" style="color:red; text-align:right"></col>
    <thead>
    …
</table>
```

但是，现代标准浏览器仅支持 background-color 和 width 属性，不支持其他 CSS 属性。IE 诡异模式支持所有 CSS 属性。

第 7 步，设计隔行换色样式。

隔行换色是一款比较经典的表格样式，这种样式主要是从用户体验角度来设计的，以提升用户浏览数据的速度和准确度。隔行换色的传统设计方法：定义一个类，然后把该类应用到所有奇数行或偶数行。推荐方法：使用 CSS3 选择器匹配表格中偶数行和奇数行，并设计不同的样式。

```
tbody tr:nth-child(odd) td {
    background-color: #CEA;
```



```
        border-bottom: 2px solid #67BD2A;
}
```

第 8 步，设计鼠标指针移过数据行时的交互样式。

```
tbody tr:hover td, tbody tr:hover th {
        background-color: #8b7;
        color: #fff;
}
```







视 频 讲 解

7.3.2　设计粗线框表格

本例以 7.3.1 节案例的数据表格结构为基础，介绍如何使用 CSS 把表格设计为网格化效果，以此理解 CSS 控制表格的方法，案例效果如图 7.13 所示。

【操作步骤】

第 1 步，新建 HTML5 文档，复制 7.3.1 节案例的数据表格结构，把第一列单元格<td>标签改为<th>标签。

第 2 步，规划整个页面的基本显示属性，设置表格样式。

```
body { /*网页基本样式类*/
        background-color: #f8e6e6;                /*网页背景颜色*/
        margin: 50px;                             /*表格四周补白*/
}
table {/*表格样式*/
        border: 6px double #3186dd;               /*表格边框*/
        font-family: Arial;
        text-align: center;                       /*表格中文字水平居中对齐*/
        border-collapse: collapse;                /*边框重叠*/
}
```

此时显示效果如图 7.14 所示。可以看到，网页背景颜色发生了改变，且表格显示了边框。



历届奥运会中国奖牌数

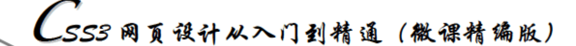
编号	年份	城市	金牌	银牌	铜牌	总计
第23届	1984年	洛杉矶（美国）	15	8	9	32
第24届	1988年	汉城（韩国）	5	11	12	28
第25届	1992年	巴塞罗那（西班牙）	16	22	16	54
第26届	1996年	亚特兰大（美国）	16	22	12	50
第27届	2000年	悉尼（澳大利亚）	28	16	15	59
第28届	2004年	雅典（希腊）	32	17	14	63
第29届	2008年	北京（中国）	51	21	28	100
第30届	2012年	伦敦（英国）	38	27	23	88
第31届	2016年	里约热内卢（巴西）	26	18	26	70
合计	544枚					

图 7.13　设计粗线框表格

图 7.14　设置网页基本属性及表格样式

第 3 步，设置表格标题的样式。

```
table caption   {                                 /*设置表格标题*/
        padding-top: 3px;                         /*设置表格标题的顶部边距*/
```





```
    padding-bottom: 4px;                /*设置表格标题的底部边距*/
    font-size: 30px;                    /*表格标题字体大小*/
    color: red;                         /*表格标题字体颜色*/
}
```



第 4 步，设置表格中的单元格样式。

```
table th {/*表格的行、列名称单元格的样式*/
    border: 2px solid #429fff;          /*行、列名称边框*/
    background-color: #d2e8ff;          /*行、列名称单元格的背景颜色*/
    font-weight: bold;                  /*行、列名称字体加粗*/
    padding-top: 4px;                   /*设置行、列名称单元格的上、下、左、右边距*/
    padding-bottom: 4px;
    padding-left: 10px;
    padding-right: 10px;
}
table td {/*表格单元格样式*/
    border: 2px solid #429fff;          /*单元格边框*/
}
```

以上代码中，分别设置了<th>和<td>标签的样式，对表格的单元格进行了背景颜色、边框样式的设置，从而达到美化表格的目的。至此，整个案例设计完成。



7.3.3 设计浅色风格表格

本例通过浅色搭配设计一款淡雅的表格效果，同时设置表格隔行变色，使奇数行和偶数行背景颜色不同，让数据行看起来清晰、明了，案例效果如图 7.15 所示。



历届奥运会中国奖牌数

编号	年份	城市	金牌	银牌	铜牌	总计
第23届	1984年	洛杉矶（美国）	15	8	9	32
第24届	1988年	汉城（韩国）	5	11	12	28
第25届	1992年	巴塞罗那（西班牙）	16	22	16	54
第26届	1996年	亚特兰大（美国）	16	22	12	50
第27届	2000年	悉尼（澳大利亚）	28	16	15	59
第28届	2004年	雅典（希腊）	32	17	14	63
第29届	2008年	北京（中国）	51	21	28	100
第30届	2012年	伦敦（英国）	38	27	23	88
第31届	2016年	里约热内卢（巴西）	26	18	26	70
合计	544枚					

图 7.15　设计浅色风格表格

【操作步骤】

第 1 步，新建 HTML5 文档，复制 7.3.1 节案例的数据表格结构。

第 2 步，定义网页基本属性、表格<table id="mytable">的样式，以及表格标题样式。

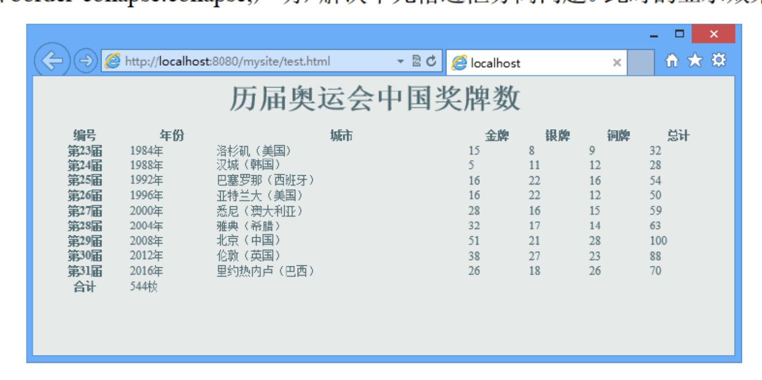
```
body { /*网页基本样式*/
    background: #E6EAE9;
}
table { /*表格样式*/
    width: 700px;                       /*表格宽度*/
```



```
        padding: 0;
        margin: 0;
        border: 1px solid #C1DAD7;              /*表格边框*/
        border-collapse: collapse;
    }
    caption { /*设置表格标题*/
        padding: 0 0 5px 0;
        text-align: center;                     /*水平居中*/
        font-size: 30px;                        /*字体大小*/
        font-weight: bold;                      /*字体加粗*/
    }
```



在以上代码中，首先定义了页面的背景颜色，然后设置了表格的宽度为 700px，并为其添加了表格边框。为 table 添加 border-collapse:collapse;声明，解决单元格边框分离问题。此时的显示效果如图 7.16 所示。



历届奥运会中国奖牌数

编号	年份	城市	金牌	银牌	铜牌	总计
第23届	1984年	洛杉矶（美国）	15	8	9	32
第24届	1988年	汉城（韩国）	5	11	12	28
第25届	1992年	巴塞罗那（西班牙）	16	22	16	54
第26届	1996年	亚特兰大（美国）	16	22	12	50
第27届	2000年	悉尼（澳大利亚）	28	16	15	59
第28届	2004年	雅典（希腊）	32	17	14	63
第29届	2008年	北京（中国）	51	21	28	100
第30届	2012年	伦敦（英国）	38	27	23	88
第31届	2016年	里约热内卢（巴西）	26	18	26	70
合计	544枚					

图 7.16　设置表格基本属性

第 3 步，使用 thead th 选择器单独为列标题行定义样式，使用 tbody 选择器定义数据区域背景色。

```
    thead th {
        color: #4f6b72;
        border: 1px solid #C1DAD7;
        letter-spacing: 2px;
        text-align: left;
        padding: 6px 6px 6px 12px;
        background: #CAE8EA;
    }
    tbody { background: #fff;}
```

第 4 步，使用 tbody th, tbody td 组合选择器为数据区单元格定义样式，这样就避免了为每个单元格引用类样式。

```
    tbody th, tbody td {
        border: 1px solid #C1DAD7;
        font-size: 14px;
        padding: 6px 6px 6px 12px;
        color: #4f6b72;
    }
```

第 5 步，使用 CSS3 的结构伪类选择器 tbody tr:nth-child(2n)专门为数据区域内所有偶数行定义特殊样式，实现隔行换色效果，这样避免了单独为偶数行单元格应用特殊类样式。



```
tbody tr:nth-child(2n) {
    background: #F5FAFA;
    color: #797268;
}
```



Note



视频讲解

7.3.4 设计清新风格表格

本例设计一款表格样式，其整体色调以清淡为主，边框线以淡蓝色为主色调，并配以 12px 的灰色字体，营造一种轻松的视觉效果。然后，使用隔行换色样式分行显示数据，这也是目前数据表格的主流样式，它符合视线的换行显示，避免错行阅读数据。使用渐变背景图像来设计表格列标题，使表格看起来更大方，富有立体感。

【操作步骤】

第 1 步，新建 HTML5 文档，复制 7.3.1 节案例的数据表格结构。

第 2 步，定义表格样式。表格样式包括 3 部分内容：表格边框和背景样式、表格内容显示样式和表格布局样式。布局样式包括：定义表格固定宽度解析，这样能够优化解析速度；显示空单元格；合并单元格的边框线，并设置表格居中显示。表格边框为 1px 宽的浅蓝色实线框，字体大小为 12px，颜色为灰色。

```
table {/*表格基本样式*/
    table-layout:fixed;                    /*固定表格布局，优化解析速度*/
    empty-cells:show;                      /*显示空单元格*/
    margin:0 auto;                         /*居中显示*/
    border-collapse: collapse;             /*合并单元格边框*/
    border:1px solid #cad9ea;              /*边框样式*/
    color:#666;                            /*灰色字体*/
    font-size:12px;                        /*字体大小*/
}
```

提示： table-layout 是 CSS 定义的一个标准属性，用来设置表格布局的算法，取值包括 auto 和 fixed。当取值为 auto 时，则布局将基于单元格内包含的内容来进行，表格在每一单元格内所有内容读取计算之后才会显示出来。当取值为 fixed 时，表示固定布局算法，在这种算法中，表格和列的宽度取决于 col 对象的宽度总和，如果没有指定，则根据第一行每个单元格的宽度。如果表格没有指定宽度，则表格被呈递的默认宽度为 100%。设置 auto 布局算法，需要进行两次布局计算，影响客户端的解析速度，而 fixed 布局算法仅需要一次计算，所以速度非常快。

第 3 步，定义列标题样式。列标题样式主要涉及背景图像的设计，具体代码如下：

```
th {/*列标题样式*/
    background-image: url(images/th_bg1.gif);    /*指定渐变背景图像*/
    background-repeat:repeat-x;                  /*定义水平平铺*/
    height:30px;                                 /*固定高度*/
}
```

列标题样式的设计难点是背景图像的制作，具体制作方法这里就不再详细讲解，读者可以参考本案例效果在 Photoshop 中进行设计。

第 4 步，定义单元格的显示样式。这里主要定义单元格的高度、边框线和补白。定义单元格左右两侧的补白目的是避免单元格与数据拥挤在一起。



```
td {height:20px; /*固定高度*/}                /*单元格的高度*/
td, th {/*单元格的边框线和补白*/
    border:1px solid #cad9ea;                 /*单元格边框线，应与表格边框线一致*/
    padding:0 1em 0;                          /*单元格左右两侧的补白，一个字距*/
}
```

第 5 步，定义隔行变色样式，定义比边框色稍浅的背景色。

```
tbody tr:nth-child(2n) {background-color: #f5fafe;}
```

第 6 步，保存页面，在浏览器中预览，显示效果如图 7.17 所示。

历届奥运会中国奖牌数

编号	年份	城市	金牌	银牌	铜牌	总计
第23届	1984年	洛杉矶（美国）	15	8	9	32
第24届	1988年	汉城（韩国）	5	11	12	28
第25届	1992年	巴塞罗那（西班牙）	16	22	16	54
第26届	1996年	亚特兰大（美国）	16	22	12	50
第27届	2000年	悉尼（澳大利亚）	28	16	15	59
第28届	2004年	雅典（希腊）	32	17	14	63
第29届	2008年	北京（中国）	51	21	28	100
第30届	2012年	伦敦（英国）	38	27	23	88
第31届	2016年	里约热内卢（巴西）	26	18	26	70
合计	544枚					

图 7.17　设计表格效果

7.3.5　设计圆润边框表格



视 频 讲 解

本例使用 CSS3 新技术设计圆润风格的表格效果：使用 border-radius 定义圆角；使用 box-shadow 为表格添加内阴影，设计高亮边效果；使用 transition 定义过渡动画，让鼠标指针移过数据行，渐显浅色背景；使用 linear-gradient()函数定义标题列渐变背景效果，以替换传统使用背景图像模拟渐变效果；使用 text-shadow 属性定义文本阴影，让标题文本看起来更富立体感。演示效果如图 7.18 所示。



历届奥运会中国奖牌数

编号	年份	城市	金牌	银牌	铜牌	总计
第23届	1984年	洛杉矶（美国）	15	8	9	32
第24届	1988年	汉城（韩国）	5	11	12	28
第25届	1992年	巴塞罗那（西班牙）	16	22	16	54
第26届	1996年	亚特兰大（美国）	16	22	12	50
第27届	2000年	悉尼（澳大利亚）	28	16	15	59
第28届	2004年	雅典（希腊）	32	17	14	63
第29届	2008年	北京（中国）	51	21	28	100
第30届	2012年	伦敦（英国）	38	27	23	88
第31届	2016年	里约热内卢（巴西）	26	18	26	70
合计	544枚					

图 7.18　设计表格样式

【操作步骤】

第 1 步，新建 HTML5 文档，复制 7.3.1 节案例的数据表格结构。





Note

第 2 步，在头部区域<head>标签中插入一个<style type="text/css">标签，在该标签中输入下面样式代码，定义表格默认样式，并定义表格外框主题类样式。

```
table {
    *border-collapse: collapse; /*兼容 IE7 及其以下版本浏览器*/
    border-spacing: 0;
    width: 100%;}
.bordered {
    border: solid #ccc 1px;
    border-radius: 6px;
    box-shadow: 0 1px 1px #ccc;}
```

第 3 步，输入下面样式代码，统一单元格样式，定义边框、空隙效果。

```
.bordered td,  .bordered th {
    border-left: 1px solid #ccc;
    border-top: 1px solid #ccc;
    padding: 10px;
    text-align: left;}
```

第 4 步，输入下面样式代码，设计表格标题列样式，通过渐变效果设计标题列背景效果，并适当添加阴影，营造立体效果。

```
.bordered th {
    background-color: #dce9f9;
    background-image: linear-gradient(top, #ebf3fc, #dce9f9);
    box-shadow: 0 1px 0 rgba(255,255,255,.8) inset;
    border-top: none;
    text-shadow: 0 1px 0 rgba(255,255,255,.5);}
```

第 5 步，输入下面样式代码，设计圆角效果。在制作表格圆角效果之前，有必要先完成这一步。表格的 border-collapse 默认值是 separate，将其值设置为 0，也就是“border-spacing:0;”。为了能兼容 IE7 以及更低版本的浏览器，需要加上一个特殊的属性 border-collapse，并且将其值设置为 collapse。

```
table {
    *border-collapse: collapse; /*兼容 IE7 及其以下版本浏览器*/
    border-spacing: 0;
}
```

第 6 步，设计圆角效果，具体代码如下：

```
/*==整个表格设置了边框，并设置了圆角==*/
.bordered {border: solid #ccc 1px; border-radius: 6px;}
/*==表格头部第一个 th 需要设置一个左上角圆角==*/
.bordered th:first-child {border-radius: 6px 0 0 0;}
/*==表格头部最后一个 th 需要设置一个右上角圆角==*/
.bordered th:last-child {border-radius: 0 6px 0 0;}
/*==表格最后一行的第一个 td 需要设置一个左下角圆角==*/
.bordered tr:last-child td:first-child {border-radius: 0 0 0 6px;}
/*==表格最后一行的最后一个 td 需要设置一个右下角圆角==*/
.bordered tr:last-child td:last-child {border-radius: 0 0 6px 0;}
```

第 7 步，由于在 table 中设置了一个边框，为了显示圆角效果，需要在表格的 4 个角的单元格上分别设置圆角效果，并且其圆角效果需要和表格的圆角值大小一样。反之，如果在 table 上没有设置



边框，只需要在表格的 4 个角落的单元格设置圆角，就能实现圆角效果。

```
/*==表格头部第一个 th 需要设置一个左上角圆角==*/
.bordered th:first-child {border-radius: 6px 0 0 0;}
/*==表格头部最后一个 th 需要设置一个右上角圆角==*/
.bordered th:last-child {border-radius: 0 6px 0 0;}
/*==表格最后一行的第一个 td 需要设置一个左下角圆角==*/
.bordered tfoot td:first-child {border-radius: 0 0 0 6px;}
/*==表格最后一行的最后一个 td 需要设置一个右下角圆角==*/
.bordered tfoot td:last-child {border-radius: 0 0 6px 0;}
```



在上面的代码中，使用了许多 CSS3 的伪类选择器。

第 8 步，除了使用 CSS3 选择器外，本案例还采用了很多 CSS3 的相关属性，这些属性将在后面章节中进行详细介绍。例如：

☑ 使用 box-shadow 制作表格的阴影。

```
.bordered {box-shadow: 0 1px 1px #ccc;}
```

☑ 使用 transition 制作 hover 过渡效果。

```
.bordered tr {transition: all 0.1s ease-in-out;}
```

☑ 使用 gradient 制作表头渐变色。

```
.bordered th {
    background-color: #dce9f9;
    background-image: linear-gradient(to top, #ebf3fc, #dce9f9);
}
```

第 9 步，使用 CSS3 的 text-shadow 来制作文字阴影效果，使用 rgba 改变颜色透明度等。

第 10 步，为<table>标签应用 bordered 类样式。

```
<table summary="历届奥运会中国奖牌数"    class="bordered">
```

7.3.6　设计数据分组表格



视 频 讲 解

本例通过树形结构来设计层次清晰的分类数据表格效果。整个表格样式设计包含 4 个技巧：

☑ 适当修改数据表格的结构，使其更利于树形结构的设计。

☑ 借助背景图像应用技巧来设计树形结构标志。

☑ 借助伪类选择器来设计鼠标经过行时变换背景颜色。

☑ 通过边框和背景色来设计列标题的立体显示效果。

【操作步骤】

第 1 步，新建 HTML5 文档，复制 7.3.1 节案例的数据表格结构。

第 2 步，修改数据表的结构。在修改数据表结构时，不要破坏数据表的基本结构，主要强化数据表格的分组。使用 thead 把标题分为一组（标题区域），使用多个 tbody 把数据分为多组（数据区域）。根据数据分类的需要，在每个 tbody 内部增加一个合并的数据行，该行仅包含一个单元格，为了避免破坏结构，使用 colspan="7"合并单元格。修改之后的数据表格结构如下。

```
<table summary="历届奥运会中国奖牌数">
    <caption>历届奥运会中国奖牌数</caption>
```






```
        <thead>
            <tr></tr>
        </thead>
        <tbody>
            <tr><td colspan="7">第一时期</td></tr>
            <tr>…</tr>
            <tr>…</tr>
            <tr>…</tr>
            <tr>…</tr>
            <tr>…</tr>
            <tr>…</tr>
        </tbody>
        <tbody>
            <tr><td colspan="7">第二时期</td></tr>
            <tr>…</tr>
            <tr>…</tr>
            <tr>…</tr>
        </tbody>
        <tfoot>
            <tr><th>合计</th><td colspan="6">544 枚</td></tr>
        </tfoot>
    </table>
```

第 3 步，重置基本表格对象的默认样式。例如，在 body 元素中定义页面字体类型，通过 table 元素定义数据表格的基本属性，以及其包含文本的基本显示样式。同时统一标题单元格和普通单元格的基本样式。

```
body {font-family:"宋体" arial, helvetica, sans-serif;        /*页面字体类型*/
}/*页面基本属性*/
table {/*表格基本样式*/
    border-collapse: collapse;                                /*合并单元格边框*/
    font-size: 85%;                                           /*字体大小，约为 14px*/
    line-height: 1.1;                                         /*行高，使数据显得更紧凑*/
    width: 96%;                                               /*固定宽度*/
    margin: auto;                                             /*水平居中显示*/
    border:solid 6px #c6ceda;                                 /*添加粗边框，颜色与标题行背景色一致*/
}
th {/*列标题基本样式*/
    font-weight: normal;                                      /*普通字体，不加粗显示*/
    text-align: left;                                         /*标题左对齐*/
    padding-left: 15px;                                       /*定义左侧补白*/
}
th, td {padding: .6em .6em; /*增加补白效果，避免数据拥挤在一起*/
}/*单元格基本样式*/
```

第 4 步，定义列标题的立体效果。列标题的立体效果主要借助边框样式来实现，设计顶部、左侧和右侧边框样式为 1px 宽的白色实线，而底部边框则设计为 2px 宽的浅灰色实线，这样就可以营造出一种淡淡的立体凸起效果。

```
thead th,tfoot th, tfoot td {/*列标题样式，立体效果*/
    background: #c6ceda;                                      /*背景色*/
```



```
        border-color: #fff #fff #888 #fff;                  /*设置立体边框效果*/
        border-style: solid;                                /*实线边框样式*/
        border-width: 1px 1px 2px 1px;                      /*定义边框大小*/
        padding-left: .5em;                                 /*增加左侧的补白*/
    }
```



第 5 步，定义树形结构效果。树形结构主要利用虚线背景图像（ 和 ）来模拟，借助背景图像的灵活定位特性，可以精确设计出树形结构样式。然后使用结构伪类选择器分别把它们应用到每行的第一个单元格中。

```
    tbody tr td:first-child {/*树形结构非末行图标样式*/
        background: url(images/dots.gif) 18px 54% no-repeat;        /*定义树形结构末行图标*/
        padding-left: 26px;                                         /*增加左侧的补白*/
    }
    tbody tr:last-child td:first-child {/*树形结构末行图标样式*/
        background: url(images/dots2.gif) 18px 54% no-repeat;       /*定义树形结构的末行图标*/
        padding-left: 26px;                                         /*增加左侧的补白*/
    }
```

第 6 步，为分类标题行定义一个样式类。通过为该行增加一个提示图标，以及设置行背景色，来区分不同分类行之间的视觉分类效果。

```
    tbody tr:first-child td {/*数据分类标题行的样式*/
        background:#eee url(images/arrow.gif) no-repeat 12px 50%;   /*背景图像，定义提示图标*/
        padding-left: 28px;                                         /*增加左侧的补白*/
        font-weight:bold;                                           /*字体加粗显示*/
        color:#444;                                                 /*字体颜色*/
    }
```

第 7 步，设计当鼠标经过每行时变换背景色，以此显示当前行效果。

```
    tr:hover, td.start:hover, td.end:hover {/*鼠标经过行、单元格上时的样式*/
        background: #FF9;                                           /*变换背景色*/
    }
```

第 8 步，保存页面，在浏览器中预览，显示效果如图 7.19 所示。

历届奥运会中国奖牌数

编号	年份	城市	金牌	银牌	铜牌	总计
第一时期						
第23届	1984年	洛杉矶（美国）	15	8	9	32
第24届	1988年	汉城（韩国）	5	11	12	28
第25届	1992年	巴塞罗那（西班牙）	16	22	16	54
第26届	1996年	亚特兰大（美国）	16	22	12	50
第27届	2000年	悉尼（澳大利亚）	28	16	15	59
第28届	2004年	雅典（希腊）	32	17	14	63
第二时期						
第29届	2008年	北京（中国）	51	21	28	100
第30届	2012年	伦敦（英国）	38	27	23	88
第31届	2016年	里约热内卢（巴西）	26	18	26	70
合计	544枚					

图 7.19　设计数据分组表格效果



7.3.7　设计单线表格

本例在前面案例数据表格结构的基础上，使用 CSS3 技术设计一款单线表格，效果如图 7.20 所示。



历届奥运会中国奖牌数

编号	年份	城市	金牌	银牌	铜牌	总计
第23届	1984年	洛杉矶（美国）	15	8	9	32
第24届	1988年	汉城（韩国）	5	11	12	28
第25届	1992年	巴塞罗那（西班牙）	16	22	16	54
第26届	1996年	亚特兰大（美国）	16	22	12	50
第27届	2000年	悉尼（澳大利亚）	28	16	15	59
第28届	2004年	雅典（希腊）	32	17	14	63
第29届	2008年	北京（中国）	51	21	28	100
第30届	2012年	伦敦（英国）	38	27	23	88
第31届	2016年	里约热内卢（巴西）	26	18	26	70
合计	544枚					

图 7.20　设计单线表格效果

【操作步骤】

第 1 步，新建 HTML5 文档，复制 7.3.1 节案例的数据表格结构。

第 2 步，在头部区域<head>标签中插入一个<style type="text/css">标签，在该标签中输入下面样式代码，定义表格默认样式，并定义表格外框主题类样式。

```
table {
    *border-collapse: collapse;                /*兼容 IE7 及其以下版本浏览器*/
    border-spacing: 0;
    width: 100%;
}
```

第 3 步，设计单元格和标题单元格样式，取消标题单元格的默认加粗和居中显示。

```
.table td, .table th {
    padding: 4px;                              /*增大单元格补白，避免拥挤*/
    border-bottom: 1px solid #f2f2f2;          /*定义下边框线*/
    text-align: left;                          /*文本左对齐*/
    font-weight:normal;                        /*取消加粗显示*/
}
```

第 4 步，为列标题行定义渐变背景，同时增加高亮内阴影效果，为标题文本增加淡淡阴影色。

```
.table thead th {
    text-shadow: 0 1px 1px rgba(0,0,0,.1);
    border-bottom: 1px solid #ccc;
    background-color: #eee;
    background-image: linear-gradient(to top, #f5f5f5, #eee);
}
```

第 5 步，设计数据隔行换色效果。

```
.table tbody tr:nth-child(even) {
    background: #f5f5f5;
```



```
        box-shadow: 0 1px 0 rgba(255,255,255,.8) inset;
    }
```

第 6 步，设计表格圆角效果。



```
/*左上角圆角*/
.table    thead th:first-child {border-radius: 6px 0 0 0;}
/*右上角圆角*/
.table    thead    th:last-child {border-radius: 0 6px 0 0;}
/*左下角圆角*/
.table tfoot td:first-child, .table tfoot th:first-child{border-radius: 0 0 0 6px;}
/*右下角圆角*/
.table tfoot td:last-child,.table tfoot th:last-child {border-radius: 0 0 6px 0;}
```

7.3.8　设计日历表



日历表在网页中经常看到，它适合使用表格结构进行设计。本例设计一个相对比较简单的日历表，其中有当天日期状态、当天日期文字说明，双休日以红色文字、浅灰色背景显示，周日到周一的标题加粗显示。

【操作步骤】

第 1 步，启动 Dreamweaver，新建网页，保存为 index.html，在<body>标签内输入以下代码。

```
<table>
    <caption>2017 年 7 月 1 日</caption>
    <thead>
        <tr>
            <th>日</th>
            <th>一</th>
            <th>二</th>
            <th>三</th>
            <th>四</th>
            <th>五</th>
            <th>六</th>
        </tr>
    </thead>
    <tbody>
<tr><td>29</td><td>30</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr>
<tr><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td></tr>
<tr><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td></tr>
<tr><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td></tr>
<tr><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>1</td><td>2</td></tr>
<tr><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr>
    </tbody>
</table>
```

日历表以表格结构形式表示，不仅在结构上表达了日历是一种数据型结构，而且能更显著地在页面无 CSS 样式情况下表现日历表应该具有的结构，如图 7.21 所示。

第 2 步，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后输入下面样式，设计表格框样式。



```
table {/*定义表格文字样式*/
    border-collapse:collapse; /*合并单元格之间的边*/
    border:1px solid #DCDCDC;
    font:normal 12px/1.5em Arial, Verdana, Lucida, Helvetica, sans-serif;
}
```



Note

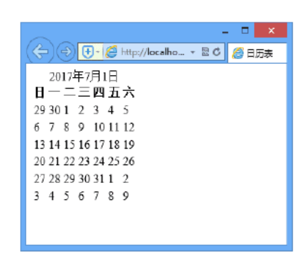


图 7.21　无 CSS 样式的日历表

合并表格单元格之间的边框，设计表格内对象的继承样式。例如，单元格之间的边框合并、文字样式。考虑日历表中显示的内容以数字居多，因此文字主要采用了英文字体。

第 3 步，设计表格标题样式。设置表头的高度属性和文字颜色。

```
caption { /*定义表头的样式，文字居中等*/
    text-align:center;
    line-height:46px;
    font-size:20px;
    color: blue;
}
```

第 4 步，设计单元格基本样式。

```
td, th {/*将单元格内容和单元格标题的共同点归为一组样式定义*/
    width: 40px;
    height: 40px;
    text-align: center;
    border: 1px solid #DCDCDC;
}
th {/*针对单元格标题定义样式，使其与单元格内容产生区别*/
    color: #000000;
    background-color: #EEEEEE;
}
```

单元格内容<td>标签和单元格标题<th>标签所需要的样式只有背景颜色和文字颜色不同，因此可以将这两个元素归为一个组定义样式，然后单独针对单元格标题定义背景颜色和文字颜色。这样的处理方式不仅减少了 CSS 样式的代码，也能使 CSS 样式代码更加直观，对于后期维护也有不少帮助。

第 5 步，单元格<td>标签中所显示的时间是当前系统所显示的时间，添加一个名为 current 的 class 类名，并将其 CSS 样式定义的与其他单元格内容不同，突出显示当前日期。而且.current 类还有一个作用是为程序开发人员提供一个接口，方便他们在程序开发的过程中调用这个类名，便于判断系统当前日期后为页面实现效果。

```
td.current {/*定义当前日期的单元格内容样式*/
    font-weight:bold;
    color:#FFFFFF;
```



```
        background-color: blue;
}
```

第 6 步，设计.current 类之后，把该类绑定到表格当日单元格中，如<td class="current">1</td>。

第 7 步，为了能更好地体现某个月份上一个月份的月尾几天和下一个月份的月头几天在当前月份中的位置，可以在页面中添加以下内容，并通过 CSS 样式将其视觉效果弱化。



```
td.last_month, td.next_month {color:#DFDFDF;} /*定义上个月和下个月日期在当前月中的文字颜色*/
```

第 8 步，设计.last_month 和.next_month 类之后，把这两个类绑定到表格非当月单元格中，代码如下。

```
<tr>
        <td class="last_month">29</td>
        <td class="last_month">30</td>
        <td class="current">1</td>
        <td>2</td>
        <td>3</td>
        <td>4</td>
        <td>5</td>
</tr>
…
<tr>
        <td class="next_month">3</td>
        <td class="next_month">4</td>
        <td class="next_month">5</td>
        <td class="next_month">6</td>
        <td class="next_month">7</td>
        <td class="next_month">8</td>
        <td class="next_month">9</td>
</tr>
```

第 9 步，设计表格列组样式。在表格框<table>内部前面添加如下代码：

```
<table>
        <caption> 2017 年 7 月 1 日</caption>
        <colgroup span="7">
        <col span="1" class="day_off">
        <col span="5">
        <col span="1" class="day_off">
        </colgroup>
        <thead>
…
```

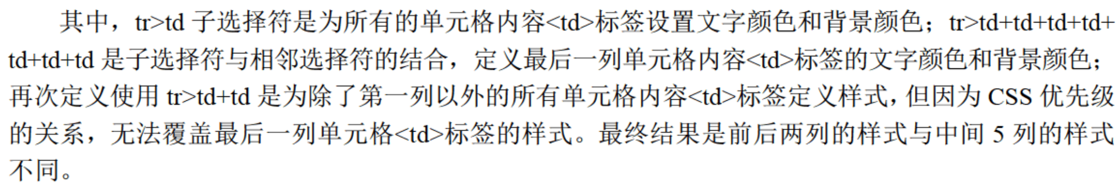
第 10 步，使用<colgroup>标签将表格前后两列（即双休日）的日期定义为一种样式，以区别于其他单元格内容中的日期。

```
tr>td, tr>td+td+td+td+td+td+td {
                color:#B3222B;
                background-color:#F8F8F8;
} /*定义第一列和最后一列单元格内容（即双休日）的样式*/
tr>td+td {
```



```
            color:#333333;
            background-color:#FFFFFF;
} /*定义中间 5 列单元格内容的样式*/
col.day_off {
            color:#B3222B;
            background-color:#F8F8F8;
} /*针对 IE 浏览器定义双休日的单元格样式*/
```

Note



其中，tr>td 子选择符是为所有的单元格内容<td>标签设置文字颜色和背景颜色；tr>td+td+td+td+td+td+td 是子选择符与相邻选择符的结合，定义最后一列单元格内容<td>标签的文字颜色和背景颜色；再次定义使用 tr>td+td 是为除了第一列以外的所有单元格内容<td>标签定义样式，但因为 CSS 优先级的关系，无法覆盖最后一列单元格<td>标签的样式。最终结果是前后两列的样式与中间 5 列的样式不同。

col.day_off 是针对 IE 浏览器而定义样式，主要是第一列与最后一列的文字颜色和背景颜色。该选择符的定义方式需要 XHTML 结构的支持，读者可以查看 XHTML 结构中<col>标签选择控制列的方式。

第 11 步，设计完毕，保存页面，在浏览器中预览，显示效果如图 7.22 所示。

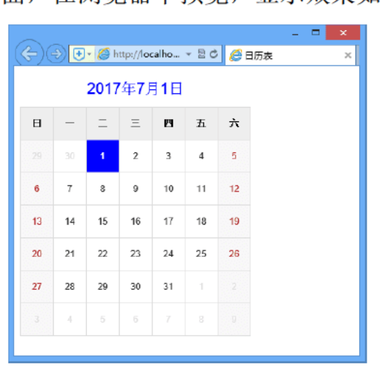


图 7.22　日历表页面设计效果

7.4　在线练习

本节通过大量的上机示例，帮助初学者练习使用 HTML5 设计表格结构和样式。感兴趣的读者可以扫码练习：（1）表格结构；（2）表格美化。



在线练习 1



在线练习 2



第8章 使用 CSS 美化表单

表单是网页交互的基础，网站一般都借助表单，如注册表、登录表、调查表和留言表等，实现用户与服务器之间的信息交流。HTML5 新增了很多表单控件，完善了部分表单控件的功能，新特性提供了更好的用户体验和输入控制。

【学习重点】

- 正确使用各种表单控件。
- 掌握表单属性的设置。
- 设计易用性表单页面。


视频讲解


Note

8.1 HTML5 表单基础

与表格一样，表单也包含多个标签，它由很多控件构成，如文本框、文本区域、单选按钮、复选框、下拉菜单和按钮等。一般情况下，表单结构可分为以下 3 部分。

☑ 表单框：使用<form>标签定义，主要功能为定义提交表单的处理方法、URL 和字符编码等。

☑ 表单对象：包括文本框、密码框、隐藏域、多行文本框、复选框、单选按钮、下拉选择框、文件上传框、提交按钮、复位按钮和一般按钮等。

☑ 辅助对象：包括提示性标签<label>、表单对象分组标签<fieldset>，用于表单结构的辅助设计。

【示例】新建网页，保存为 test.html，在<body>内使用<form>标签包含两个<input>标签和一个提交按钮，并借助<p>标签把按钮和文本框分行显示。

```
<form action="#" method="get" id="form1" name="form1">
    <p>用户名：<input name="" type="text" /></p>
    <p>密码：<input name="" type="text" /></p>
    <p><input type="submit" value="提交"/></p>
</form>
```

在 IE 浏览器中预览，则演示效果如图 8.1 所示。

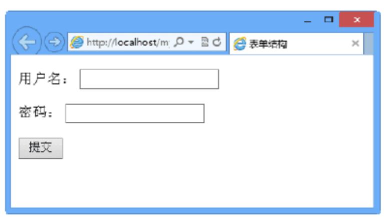


图 8.1　表单的基本效果

HTML5 定义了多个表单标签，简单说明如表 8.1 所示。

表 8.1　HTML 表单标签

标　　签	说　　明
<form>	定义供用户输入的 HTML 表单
<input>	定义输入控件
<textarea>	定义多行的文本输入控件
<button>	定义按钮
<select>	定义选择列表（下拉列表）
<optgroup>	定义选择列表中相关选项的组合
<option>	定义选择列表中的选项
<label>	定义 input 元素的标注
<fieldset>	定义围绕表单中元素的边框
<legend>	定义 fieldset 元素的标题
<isindex>	定义与文档相关的可搜索索引。不赞成使用
<datalist>	HTML5 新增标签，定义下拉列表
<keygen>	HTML5 新增标签，定义生成密钥
<output>	HTML5 新增标签，定义输出的一些类型

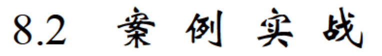


<input>标签是通用输入型表单对象，使用它可以定义多种类型的表单对象，另外 HTML5 又扩展了很多输入型表单对象，简单说明如表 8.2 所示。



表 8.2　<input>标签可定义的输入型表单对象

表 单 对 象	说　　明
<input type="text">	单行文本输入框
<input type="password">	密码输入框（输入的文字用点号表示）
<input type="checkbox">	复选框
<input type="radio">	单选按钮
<input type="file">	文件域
<input type="submit">	将表单（form）里的信息提交给表单属性 action 所指向的文件
<input type="reset">	将表单（form）里的信息清空，重新填写
<input type="color">	HTML5 新增对象，颜色选择器
<input type="date">	HTML5 新增对象，日期选择器
<input type="time">	HTML5 新增对象，时间选择器
<input type="datetime">	HTML5 新增对象，UTC 日期时间选择器
<input type="datetime-local">	HTML5 新增对象，本地日期时间选择器
<input type="week">	HTML5 新增对象，选择第几周的文本框
<input type="month">	HTML5 新增对象，月份选择器
<input type="email">	HTML5 新增对象，Email 输入框
<input type="tel">	HTML5 新增对象，电话号码输入框
<input type="url">	HTML5 新增对象，URL 输入框
<input type="number">	HTML5 新增对象，只能输入数字的文本框
<input type="range">	HTML5 新增对象，拖动条或滑块
<input type="search">	HTML5 新增对象，搜索文本框，与“type="text";”的文本框没有太大区别



8.2 案例实战

CSS 没有为表单定义专用属性，不过可以使用 CSS 其他属性，如字体、背景、颜色、边框、边距等来设计表单样式。注意，由于部分表单对象是相对复杂的控件，如下拉菜单、文件域、复选框、单选按钮等，使用 CSS 可能无法完美控制其外观，必要时需要 JavaScript 脚本辅助实现。

8.2.1　设计登录表单



视 频 讲 解

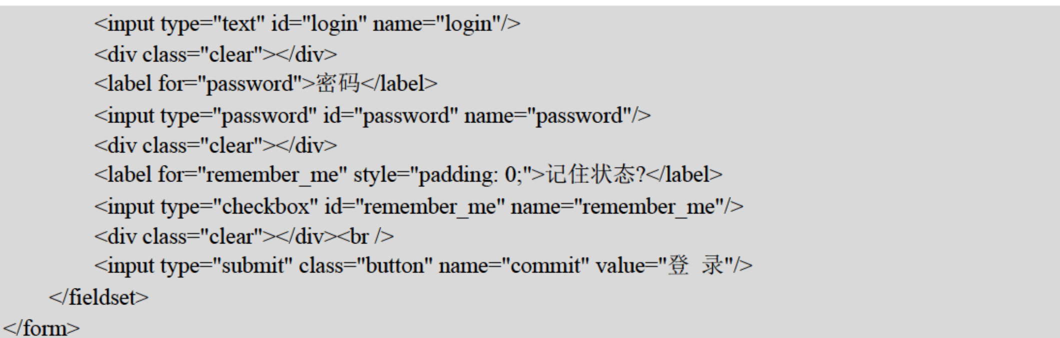
本例设计一款个性的登录表单页面，效果如图 8.2 所示。

【操作步骤】

第 1 步，新建 HTML5 文档，保存为 index.html。打开网页文档，设计如下表单结构：

```
<form id="login-form" action="#" method="post">
    <fieldset>
        <legend>登录</legend>
        <label for="login">Email</label>
```





```
            <input type="text" id="login" name="login"/>
            <div class="clear"></div>
            <label for="password">密码</label>
            <input type="password" id="password" name="password"/>
            <div class="clear"></div>
            <label for="remember_me" style="padding: 0;">记住状态?</label>
            <input type="checkbox" id="remember_me" name="remember_me"/>
            <div class="clear"></div><br />
            <input type="submit" class="button" name="commit" value="登  录"/>
        </fieldset>
    </form>
    <p align="center"><strong>&copy; www.xxxxxx.cn</strong></p>
```

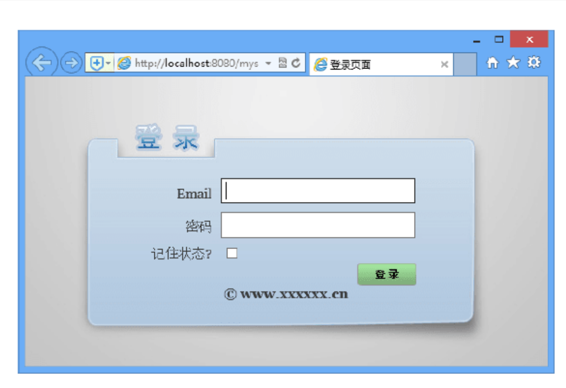


图 8.2　设计网站登录页面效果

第 2 步，为 form 元素定义 id 属性，以便对整个表单进行控制，同时方便设计 ID 样式。

第 3 步，新建 CSS 样式表文件，命名为 style.css，保存到 images 文件夹中，然后在页面头部区域导入该样式表。

```
<link rel="stylesheet" type="text/css" href="images/style.css" />
```

第 4 步，通过通配选择器清除页面中所有标签的内外边距。

```
* {margin: 0; padding: 0;}
```

第 5 步，在 body 元素中定义网页字体效果，如类型、大小和颜色。设计网页背景图像，并让背景图像偏上居中显示，禁止平铺，同时设置背景图像无法覆盖的区域颜色为浅灰色（#c4c4c4）。

```
body {font-family: Georgia, serif; background: url(login-page-bg.jpg) center -50px no-repeat #c4c4c4; color: #3a3a3a;}
```

第 6 步，定义清除样式类，以便控制页面中每个表单域的换行显示。

```
.clear {clear: both;}
```

第 7 步，设计表单对象样式。其中通过属性选择器控制复选框的样式。

```
form {width: 406px; margin: 120px auto 0;}
legend {display: none;}
fieldset {border: 0;}
label {width: 115px; text-align: right; float: left; margin: 0 10px 0 0; padding: 9px 0 0 0; font-size: 16px;}
input {width: 220px; display: block; padding: 4px; margin: 0 0 10px 0; font-size: 18px; color: #3a3a3a; font-family:
```



```
Georgia, serif;}
    input[type=checkbox] {width: 20px; margin: 0; display: inline-block;}
```

第 8 步，设计按钮在鼠标经过和未经过时的状态样式。

```
    .button {background: url(button-bg.png) repeat-x top center; border: 1px solid #999; -moz-border-radius: 5px; padding: 5px; color: black; font-weight: bold; -webkit-border-radius: 5px; font-size: 13px; width: 70px;}
    .button:hover {background: white; color: black;}
```



Note



视频讲解

8.2.2　设计信息登记表

在设计表单时，正确选用各种表单控件很重要，这是结构标准化和语义化的要求，也是用户体验的需要。例如：

☑ 不确定答案可以建议用户输入，而不是让用户选择，如姓名、地址、电话等常用信息，使用输入的方式收集会比使用选择的方式收集更加自然且简单。

☑ 对于容易记错的答案不妨让用户选择，此时就不适合让用户使用输入框来输入，如国家、年、月、日、星座等，可以使用单选按钮组、复选框、列表框、下拉菜单等形式。

☑ 当设计选择项目时，如果希望用户浏览所有选项，则应该使用单选按钮组或复选框组，而不应该使用下拉菜单。下拉菜单会隐藏部分选项，对于用户来说，可能不会耐心地逐个浏览每个菜单项。

☑ 当选项很少时，不妨考虑使用单选按钮组或复选框组，而设计过多的选项时，使用单选按钮组或复选框组会占用很大的页面，此时不妨考虑使用下拉菜单。

☑ 多项选择可以有两种设计方法：使用复选框和使用列表框。使用复选框比使用列表框更直观，而列表框的作用和操作方法不够清晰，有时还需要为其加上说明性文字，显然这样做没有使用复选框简单。

☑ 为控件设置默认值，建议采用一些提示性说明文字或常用值，能够提醒用户输入，这是一个很人性化的设计，应该考虑。

☑ 对于单选按钮组、复选框或下拉菜单，设计控件的 value 属性值或显示值时应从用户的角度考虑，努力使用户浏览选项时更方便、简单，避免出现歧义或误解的值。

☑ 对于单选、复选的选项，应减少选项的数量，同时也可以使用短语来作为选项。

☑ 对于选项的排列顺序，最好遵循合理的逻辑顺序，如按首字母排列、按声母排列等，并根据普遍情况确定默认值。

☑ 用户在设计表单时，还应该避免使用多种表单控件，使用多种表单控件能够使页面看起来更好看，但实际上不利于用户的操作。

下面介绍如何设计一个信息登记表。

【操作步骤】

第 1 步，启动 Dreamweaver，新建 HTML5 文档，保存为 index.html。

第 2 步，在页面中构建 HTML 导航框架结构。切换到代码视图，在<body>标签内输入下面代码，定义表单框架结构。

```
    <form action="#" class="form1">
        <p><em>*</em>号所在项为必填项</p>                <!--提示段落-->
        <fieldset class="fld1">                          <!--字段集 1-->
        <legend>个人信息</legend>                         <!--字段集 1 标题-->
        <ol>                                             <!--字段集 1 内嵌列表-->
```





Note

```
        <li>
            <label for="name">姓名<em>*</em></label>   <!--说明标签，以下类同-->
            <input id="name">
        </li>
        <li>
            <label for="address">地址<em>*</em></label>
            <input id="address">
        </li>
        <li>
            <label for="dob">出生<span class="sr">日</span><em>*</em></label>
            <select id="dob">
                <option value="1">1</option>
                <option value="2">2</option>
            </select>
            <label for="dob-m" class="sr">月<em>*</em></label>
            <select id="dob-m">
                <option value="1">Jan</option>
                <option value="2">Feb</option>
            </select>
            <label for="dob-y" class="sr">年<em>*</em></label>
            <select id="dob-y">
                <option value="1979">1979</option>
                <option value="1980">1980</option>
            </select>
        </li>
        <li>
            <label for="sex">性别<em>*</em></label>
            <select id="sex">
                <option value="female">女</option>
                <option value="male">男</option>
            </select>
        </li>
    </ol>
    </fieldset>
    <fieldset class="fld2">                         <!--字段集 2-->
    <legend>其他信息</legend>                       <!--字段集 2 标题-->
    <ol>                                            <!--字段集 1 内嵌列表-->
        <li>
            <fieldset>                              <!--列表内嵌字段集合-->
            <legend>你喜欢这个表单吗? <em>*</em></legend><!--子字段集合标题-->
            <label><input name="invoice-address" type="radio">喜欢</label>
            <label><input name="invoice-address" type="radio">不喜欢</label>
            </fieldset>
        </li>
        <li>
            <fieldset>
            <legend>你喜欢什么运动?</legend>
            <label for="football"><input id="football" type="checkbox">足球</label>
            <label for="golf"><input id="basketball" type="checkbox">篮球</label>
            <label for="rugby"><input id="ping" type="checkbox">乒乓球</label>
```



```
            </fieldset>
        </li>
        <li>
            <fieldset>
            <legend>请写下你的建议? <em>*</em></legend>
            <label for="comments"><textarea id="comments" rows="7" cols="25"></textarea></label>
            </fieldset>
        </li>
    </ol>
    </fieldset>
    <input value="提交个人信息" class="submit" type="submit">
</form>
```



Note

第 3 步，在<head>标签内输入<style type="text/css">，定义一个内部样式表，然后在<style>标签内输入下面样式代码：

```
body {/*定义页面基本属性*/
    font: normal 12px "宋体", Helvetica, Verdana, Arial;}
p {/*定义段落属性*/
    margin: 10px 0;
    text-align:right;}
ul, ol, dl, li, dt, dd {/*定义列表相关元素属性*/
    list-style-type:none;                /*清除样式*/
    margin:0 0 0 1em;                    /*清除边界，并定义左边界为 1 个字宽*/
    padding:0;                           /*清除补白*/
}
form {/*定义表单域基本属性*/
    padding:2em;                         /*定义补白空隙*/
    border:solid 1px #E7F8C4;            /*定义表单域边界*/
    text-align:center;                   /*居中对齐，实现按钮居中显示*/
}
fieldset {/*定义字段集基本属性*/
    text-align:left;                     /*左对齐*/
}
legend {/*定义字段集标题属性*/
    padding: 0;                          /*清除补白*/
    margin:0;                            /*清除边界*/
    color: #000;                         /*标题颜色*/
    font-weight:bold;                    /*标题加粗显示*/
}
li legend {/*定义列表内嵌字段集标题属性*/
    font-weight:normal;                  /*清除加粗显示*/
}
input, textarea, select {/*定义表单控件基本属性*/
    margin: 0;                           /*定义边界为 0*/
    padding: 0;                          /*定义补白为 0*/
}
.sr {/*定义 label 内补充信息属性*/
    position: absolute;                  /*绝对定位*/
    left: -9999em;                       /*隐藏显示，只对机器搜索使用*/
}
```





Note

```
form.form1 {/*定义表单属性*/
    width: 370px;                                       /*定义表单宽*/
    font-size: 1.1em;                                   /*定义表单字体大小*/
    color: #333;                                        /*定义表单字体颜色*/
    background: #fff url(fieldset.gif) left bottom repeat-x;    /*定义表单背景图像*/
}
form.form1 fieldset {/*定义字段集边框属性*/
    border: none;                                       /*清除边框*/
    border-top: 1px solid #C9DCA6;                      /*显示顶部边框*/
}
form.form1 .fld1 li {/*定义字段集 1 内列表项的补白*/
    padding: 4px;}
form.form1 .fld2 li {/*定义字段集 2 内列表项的补白*/
    padding: 2px;}
li   fieldset label {/*定义内嵌字段集列表项的左补白距离*/
    padding:0 0 0 2em;}
```

第 4 步，保存文档，按 F12 键，在浏览器中预览，则效果如图 8.3 所示。



图 8.3　定义表单结构



视 频 讲 解

8.2.3　设计易用表单

表单设计应考虑用户的易用性，不恰当的表单布局，如控件摆放位置、对齐方式、标签信息与周围元素的设计都会或多或少影响用户的操作。为此，用户在设计表单布局时，不妨从下面几个角度思考，来提高自己的设计水平。

1．排列方式

根据习惯，表单控件一般使用垂直排列方式进行分布，这样能够加快视觉的移动和操作。水平排列容易使视觉移动起来很累，即使多列有规律的布局也是不可取的，人眼左右晃动操作很容易出错，如图 8.4 所示（index1.html）。





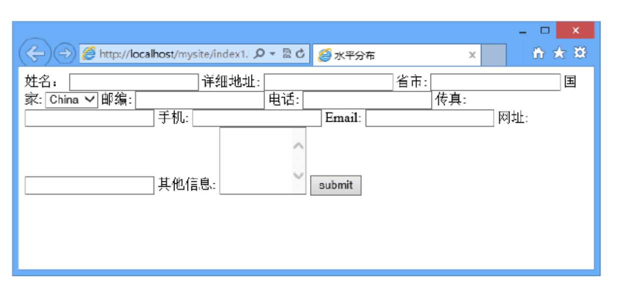


图 8.4　水平排列方式

2．控件分组

给控件分组也是表单布局中一个重要技巧，特别是表单控件很多时，分组就显得很有必要，实际上分组是帮助用户进行逻辑梳理，避免混乱。

如图 8.5 所示（index2.html），表单域由于没有实现分组，看起来很混乱，这样就会影响操作速度，每填写一项信息都需要短暂的停留，并进行思考。如果将其分为 3 组：个人信息、地址和联系信息，用户在填写表单时思路就会很清晰。

3．缩进分布

当分组标题、控件和提示信息都并列排在一起时，很容易出现主次不分的情况，如图 8.6 所示（index3.html）。用户需要分辨哪些是操作的行，哪些是说明性文字，这样会影响操作速度。对此，可以采用缩进的方式，实现多层次叠进，帮助用户快速进行阅读。

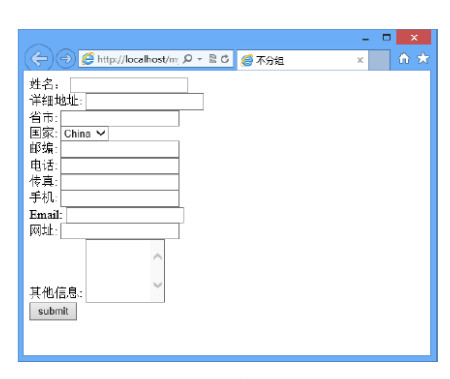


图 8.5　控件未分组

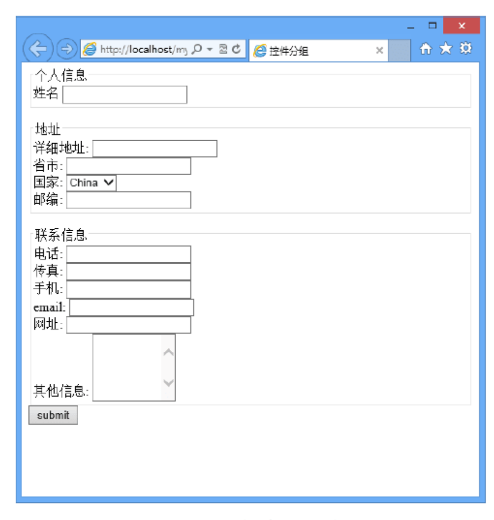


图 8.6　未缩进分布

4．标签突出

一般标签与控件水平并列分布是最佳分布方式。部分用户喜欢使用垂直分布方式，即标签在上一行，控件在下一行，这种方式对于内容较少的表单域来说可行，但如果是一个大型表单，这种方式会消耗用户的视力，降低操作速度。

在表单布局中，推荐使用加粗的标签，这可以增加它们的视觉比重，提高其显著性，如图 8.7 所示（index4.html）。否则，从用户的角度分析，标签与输入框的文字类似，可能会产生混淆的现象。



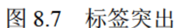


5．标签对齐

关于标签是左对齐还是右对齐问题，一般来说，一致的左对齐可以减少眼睛移动和处理时间。左对齐的标签还容易通览表单信息，用户只需要上下看看左侧标签即可，而不会被控件打断思路。但这样也容易使标签与其对应的控件之间的距离被更长的标签拉大，从而影响操作表单的时间，用户必须左右来回移动视线找到两个对应的标签和控件。

而标签右对齐布局就会避免这个问题，使得标签和控件之间的均匀分布并保持更紧密联系，如图 8.8 所示（index5.html）。当然这样分布的缺点是标签左边参差不齐的空白会影响用户快速检索表单填写内容。对此，可以根据实际情况有选择地使用标签左右对齐方式。




图 8.7　标签突出

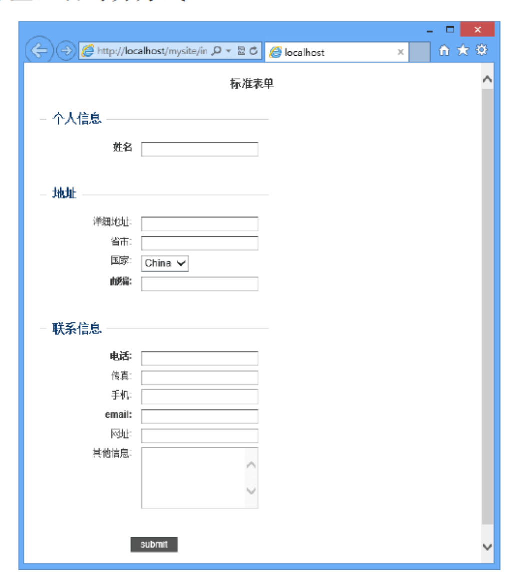


图 8.8　标签右对齐

6．背景和辅助线

上面所介绍的是一些基本的布局方法，实际上改善表单布局的方法还比较多，尝试为表单控件适当添加背景色和分隔线，通过背景色和辅助线的视觉区分，也能加快用户操作速度，这对于划分操作区信息是很有效的。

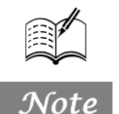
背景色和线条对于区分表单的主要操作按钮尤其有效。但在使用这些辅助元素时，要避免它们影响用户的操作，因为色彩过浓的线条和背景色都能够分散用户的视线，如图 8.9 所示（index6.html），过多的分隔线给用户阅读带来障碍。

7．动态效果

当用户选中或操作某个表单控件时，当前表单对象会显示为另一种样式，以区别于其他控件。这个技巧对于用户的操作具有提示作用，避免出现用户有时不知当前操作的是哪个表单控件的情况，如图 8.10 所示（index7.html）。

当表单控件很多时，通过添加类样式，就可以让表单更具提示性，也使用户有更好的体验。即为某个控件定义伪类，如:hover、:focus 及:focus:hover 属性样式，让输入框被鼠标激活时更加突出，利于用户集中精神填写。当然这对于老版本的 IE 浏览器没有作用，此时用户需要使用 JavaScript 脚本来控制。





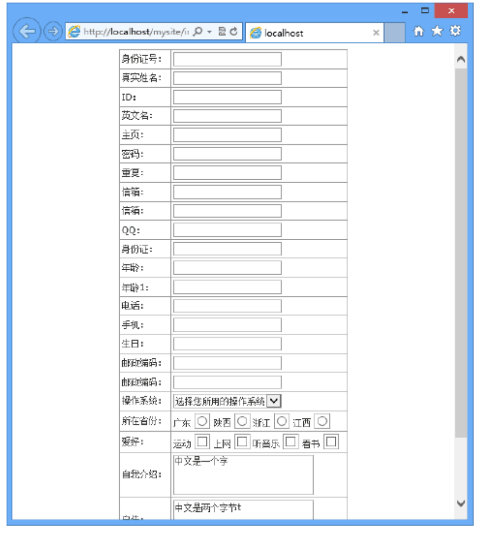


图 8.9　背景和辅助线

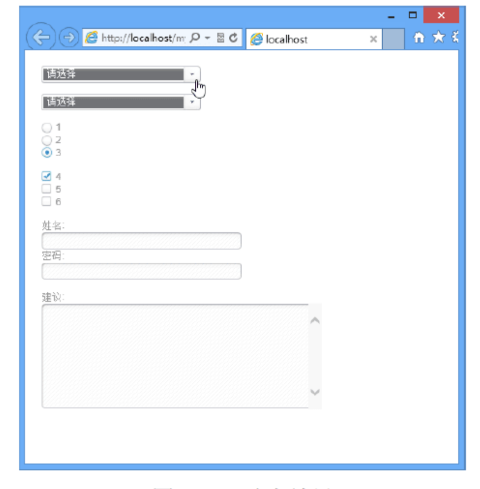


图 8.10　动态效果

8.2.4　设计注册表单



视 频 讲 解

表单设计中，一般用户喜欢设计表单对象的边框，以便实现表单与页面整体效果的融合。CSS 盒模型适用任何表单对象，所以可以使用任何盒模型属性来定义表单对象。注意，除了 form 元素是块状元素外，其他元素都以内联元素显示。

【操作步骤】

第 1 步，启动 Dreamweaver，新建 HTML5 文档，保存为 index.html。

第 2 步，在页面中构建 HTML 导航框架结构。切换到代码视图，在<body>标签内输入下面代码，定义表单框架结构。

```
<div id="box"><form  id=form1 action=#public method=post enctype=multipart/form-data>
    <h2>个人信息注册表单</h2>
    <ul>
        <li class="label">姓名
        <li> <input id=field1 size=20 name=field1>
        <li class="label">职业
        <li> <input name=field2 id=field2 size="25">
        <li class="label">详细地址
        <li><input name=field3 id=field3 size="50">
        <li class="label">邮编
        <li><input name=field4 id=field4 size="12" maxlength="12">
        <li class="label">省市
        <li><input id=field5 name=field5>
        <li class="label">国家
        <li> <select id=field6 name=field6>
                <option value=china>China</option>
                <option value=armenia>Armenia</option>
                <option value=australia>Australia</option>
                <option value=italy>Italy</option>
                <option value=japan>Japan</option>
            </select>
```





Note

```
            <li class="label">Email
            <li><input id=field7 maxlength=255 name=field11>
            <li class="label">电话
            <li><input maxlength=3 size=6 name=field8>-
                <input maxlength=8 size=16 name=field8-1>
            <li class="label"><input id=saveform type=submit value=提交></li>
        </ul>
</form> </div>
```

第 3 步，在<head>标签内输入<style type="text/css">，定义一个内部样式表，然后在<style>标签内输入下面样式代码。

```
body {
    margin: 0; padding:0;
    font-family: "lucida grande", tahoma, arial, verdana, sans-serif;}
#box{
    background:url(images/bg1.jpg);
    width:1015px; height:770px;}
#form1 {
    width:450px;
    text-align:left; font-size:12px;
    padding:12px 32px; margin:0 auto;}
#form1 h2 {
    border-bottom:dotted 1px #E37EA6;
    text-align:center;
    font-weight:normal;            /*清除标题加粗默认样式*/
}
ul {/*清除列表样式*/
    padding:0; margin:0;
    list-style-type:none;}
input {border:groove #ccc 1px;}    /*定义 3D 凹槽立体效果*/
.field6 {color:#666; width:32px;}
.label {
    font-size:13px; font-weight:bold;
    margin-top:0.7em;}
```

第 4 步，保存文档，按 F12 键，在浏览器中预览，则效果如图 8.11 所示。

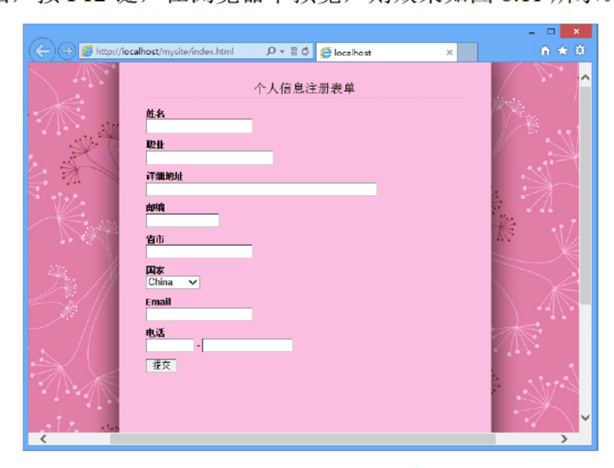


图 8.11　设计注册表单



视频讲解



Note

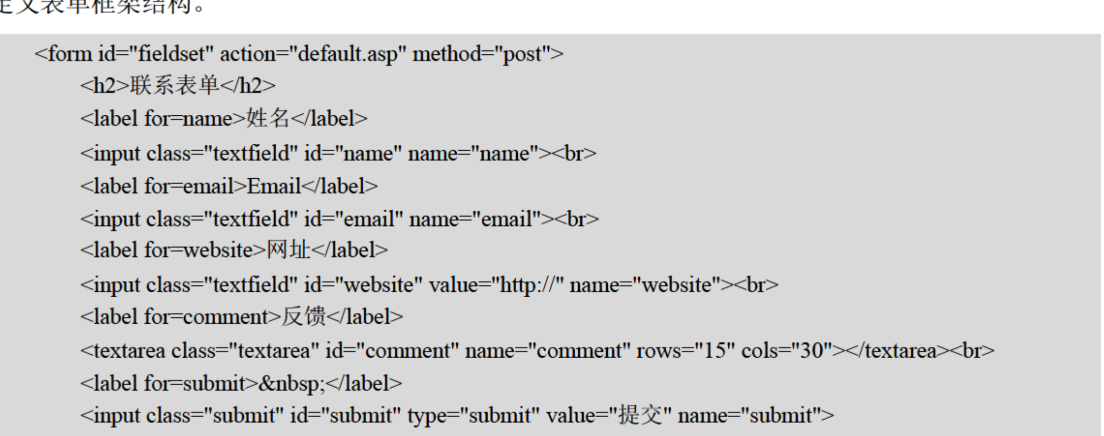
8.2.5　设计联系表单

本例设计一个联系表单，使用 CSS 背景图像来设计表单样式，使其更具艺术化。

【操作步骤】

第 1 步，启动 Dreamweaver，新建 HTML5 文档，保存为 index.html。

第 2 步，在页面中构建 HTML 导航框架结构。切换到代码视图，在<body>标签内输入下面代码，定义表单框架结构。



```
<form id="fieldset" action="default.asp" method="post">
    <h2>联系表单</h2>
    <label for=name>姓名</label>
    <input class="textfield" id="name" name="name"><br>
    <label for=email>Email</label>
    <input class="textfield" id="email" name="email"><br>
    <label for=website>网址</label>
    <input class="textfield" id="website" value="http://" name="website"><br>
    <label for=comment>反馈</label>
    <textarea class="textarea" id="comment" name="comment" rows="15" cols="30"></textarea><br>
    <label for=submit> </label>
    <input class="submit" id="submit" type="submit" value="提交" name="submit">
</form>
```

第 3 步，在<head>标签内输入<style type="text/css">，定义一个内部样式表，然后在<style>标签内输入下面样式代码。

```
body {/*定义页面属性*/
    font-size: 12px;                                        /*定义字体大小*/
    margin: 50px;                                           /*定义边界，避免顶部跑到页面外边*/
    color: #666;                                            /*定义颜色*/
    font-family: 宋体, verdana, arial, helvetica, sans-serif;  /*定义字体*/
}
#fieldset {/*定义表单属性*/
    border: #fff 0px solid;                                 /*清除边框*/
    width: 300px;                                           /*定义表单域宽度*/
    background-color: #ccc;                                 /*定义浅灰色背景*/
}
#fieldset h2 {/*定义表单标题属性*/
    padding: 0.2em;                                         /*定义补白，增加边缘空隙*/
    margin:0;                                               /*清除标题预定义边界*/
    position: relative;                                     /*相对定位*/
    top: -1em;                                              /*在现有流位置向上移动一个字体距离*/
    background: url(h2_bg.gif) no-repeat;                   /*定义背景图像，圆角显示*/
    width: 194px;                                           /*定义宽度，该宽度与背景图像宽相同*/
    font-size: 2em;                                         /*定义字体大小*/
    color: #fff;                                            /*定义字体颜色*/
    white-space: pre;                                       /*保留标题预定义格式，可以保留多行显示*/
```





Note

```
        letter-spacing: -1px;                                  /*收缩字距*/
        text-align:center;                                     /*居中显示*/
    }
    #fieldset label {/*定义表单标签属性*/
        padding: 0.2em;                                        /*增加边距空隙*/
        margin: 0.4em 0px 0px;                                 /*增加顶部边界，加大与上一个控件的间距*/
        float: left;                                           /*向左浮动*/
        width: 70px;                                           /*定义宽度*/
        text-align: right;                                     /*右对齐*/
    }
    .br {/*隐藏换行标签，也不占据位置*/
        display: none;}
    .textfield {/*定义输入表单控件*/
        border: #fff 0px solid;                                /*清除边框*/
        padding: 3px 8px;                                      /*增加内容边距空隙*/
        margin: 3px;                                           /*定义边界距离*/
        width: 187px;                                          /*定义宽*/
        height: 20px;                                          /*定义高*/
        background: url(textfield_bg.gif) no-repeat;           /*定义输入表单控件背景图像*/
        color: #FF00FF;                                        /*定义表单显示字体颜色*/
        font: 1.1em verdana, arial, helvetica, sans-serif;     /*定义字体属性*/
    }
    textarea {/*定义文本域控件属性*/
        border: #fff 0px solid;                                /*清除边框*/
        padding:4px 8px;                                       /*增加内容边距，避免内部文本顶到边框边*/
        margin: 3px;                                           /*定义边界距离*/
        height: 150px;                                         /*定义高*/
        width: 190px;                                          /*定义宽*/
        background: url(textarea_bg.gif) no-repeat;            /*定义文本域表单控件背景图像*/
        color: #FF00FF;                                        /*定义表单显示字体颜色*/
        font: 1.1em verdana, arial, helvetica, sans-serif;     /*定义字体属性*/
    }
    .submit {/*定义按钮控件属性*/
        border: #fff 0px solid;                                /*清除边框*/
        margin: 6px;                                           /*定义边界距离 *
        width: 80px;                                           /*定义宽*/
        height: 20px;                                          /*定义高*/
        background: url(submit.gif) no-repeat;                 /*定义按钮控件背景图像*/
        text-transform: uppercase;                             /*英文大写显示*/
        font: 1.1em verdana, arial, helvetica, sans-serif;     /*定义字体属性*/
        color: #666;                                           /*定义字体颜色*/
    }
```

第 4 步，保存文档，按 F12 键，在浏览器中预览，则效果如图 8.12 所示。

关于背景图像的应用还是比较灵活的，用户可以充分发挥想象力，设计出更具创意的表单效果。例如，要制作动态表单，先制作好动态的 GIF 图像，然后引入即可。





图 8.12　设计联系表单



8.2.6　设计高亮样式

本例借助 CSS 的 UI 伪类选择器设计在表单对象获取焦点和失去焦点时，分别为该表单对象应用不同的样式，从而实现当前表单高亮显示效果，如图 8.13 所示。



视频讲解

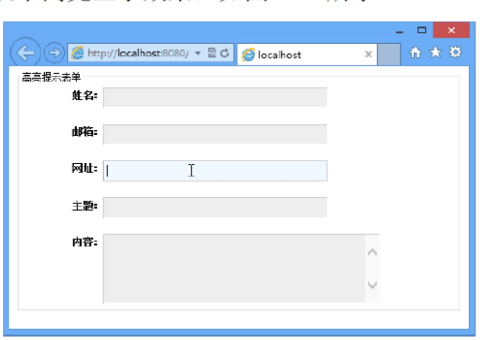
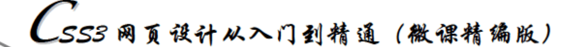


图 8.13　高亮提示表单效果

【操作步骤】

第 1 步，新建文档，在<body>标签中输入下面代码，构建一个表单结构，该结构是在 8.2.5 节示例基础上重新定义的。

```
<form id="form1" name="form1" method="post" action="">
    <fieldset>
    <legend>高亮提示表单</legend>
    <label for="name" class="title">姓名：</label>
    <input size="40" name="name" class="input1">
    <label for="email" class="title">邮箱：</label>
    <input size="40" name="email" class="input1">
    <label for="url" class="title">网址：</label>
    <input size="40" name="url" class="input1">
    <label for="subject" class="title">主题：</label>
```







Note

```
        <input size="40" name="subject" class="input1">
        <label for="message" class="title">内容：</label>
        <textarea name="message" cols="39" rows="5" class="input1" ></textarea>
        </fieldset>
</form>
```

第 2 步，在<head>标签内插入<style>标签，定义内部样式表。

第 3 步，定义 CSS 样式控制表单的显示，同时定义两个类，以供 JavaScript 属性事件调用。

```
body {/*统一网页字体大小*/
        font-size:12px;                         /*字体大小 12px*/
}
.title {/*提示文本样式类*/
        width:100px;                            /*固定宽度*/
        float:left;                             /*向左浮动定位*/
        text-align:right;                       /*文本右对齐*/
        font-weight:bold;                       /*字体加粗*/
        margin:6px 0;                           /*定义提示文本的外边距*/
}
input, textarea {/*表单对象默认显示样式*/
        background-color: #EEEEEE;
        border-bottom: #FFFFFF 1px solid;
        border-left: #CCCCCC 1px solid;
        border-right: #FFFFFF 1px solid;
        border-top: #CCCCCC 1px solid;
}
input:focus, textarea:focus {/*表单获取焦点时的样式*/
        background-color:#F0F8FF;
        border: 1px solid #999;
}
```



视 频 讲 解

8.2.7　设计图标表单

本例介绍如何把一个外部图标固定在文本框的左侧，既能点缀表单，又能够提示用户输入，演示效果如图 8.14 所示。

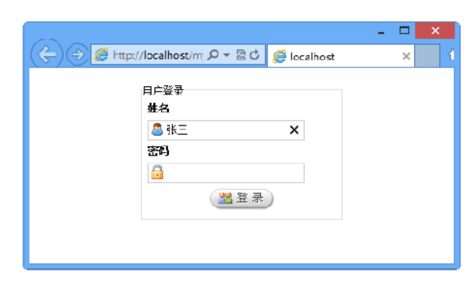


图 8.14　图标样式的表单效果

【操作步骤】

第 1 步，新建文档，在<body>标签中输入下面代码，构建一个表单结构。该结构依然保留前面示例的结构雏形，并适当进行增删。






```
<div id="login">
    <fieldset>
    <legend>用户登录</legend>
    <form action="" method="POST" class="form">
        <label for="name">姓名</label>
        <div><input name="name" type="text"  id="name" value="" /></div>
        <label for="password">密码</label>
        <div><input name="password" type="text" id="password" value="" /></div>
        <div class="button_div"><input type="image" src="images/login.gif" /></div>
    </form>
    </fieldset>
</div>
```

第 2 步，在<head>标签内插入<style>标签，定义内部样式表。

第 3 步，输入下面样式代码，使用 CSS 对这个表单进行布局。

```
* {/*清除所有元素的边距*/
    margin:0;                          /*清除内边距*/
    padding:0;                         /*清除外边距*/
}
body {/*定义网页基本属性*/
    text-align:center;                 /*网页居中显示*/
}
#login {/*表单包含框样式*/
    margin:10px auto 10px;             /*网页居中显示*/
    text-align:left;                   /*文本左对齐*/
}
fieldset {/*表单域样式*/
    width:230px;                       /*固定表单的宽度*/
    margin:20px auto;                  /*定义表单的外边距*/
    font-size:12px;                    /*统一表单的字体大小*/
}
```

第 4 步，需要让表单对象的提示文本与表单对象上下排列显示，故定义标签元素为块状元素，并定义它们的宽度为固定显示。详细代码如下：

```
label {/*定义标签提示文本的样式*/
    width:200px;                       /*固定宽度*/
    height:26px;                       /*固定高度*/
    line-height:26px;                  /*固定行高*/
    text-indent:6px;                   /*文本首行缩进 6px*/
    display:block;                     /*块状显示*/
    font-weight:bold;                  /*加粗提示文本*/
}
```

第 5 步，定义表单对象的样式。先利用分组统一所有表单对象的样式，然后利用插入背景图像的方法单独为每个文本框左侧定义一个图标。为了避免文本框内的文本遮盖背景图像图标，需要定义左侧内边距以挤出一个空间给背景图像留用。详细代码如下：

```
#name, #password {/*统一表单对象的样式*/
    border:1px solid #ccc;             /*定义表单对象的边框样式*/
    width:160px;                       /*固定宽度*/
```





Note

```
        height:22px;                                      /*固定高度*/
        margin-left:6px;                                  /*定义左侧外边距*/
        padding-left:20px;                                /*定义左侧内边距，挤出定义背景图像的空间*/
        line-height:20px;                                 /*定义行高*/
    }
    #name {/*用户名文本框图标样式*/
        background:url(images/name.gif) no-repeat 2px 2px;        /*定义用户名文本框图标*/
    }
    #password {/*用户密码文本框图标样式*/
        background:url(images/password.gif) no-repeat 2px 2px;    /*定义密码文本框图标*/
    }
    .button_div {/*按钮样式*/
        text-align:center;                                /*按钮文本居中*/
        margin:6px auto;                                  /*按钮居中显示*/
    }
```

视频讲解

8.2.8　设计反馈表

本例设计一个简单的反馈表，主要使用表单域<fieldset>标签、表单域标题<legend>标签、文件上传控件 input（type="file"）和文本域<textarea>标签。表单域<fieldset>标签主要是将表单分成多个小区域显示在网页中；表单域标题 legend>标签则是针对每个不同的表单域设置标题；文件上传控件 input（type="file"）结合后台开发程序或者 JavaScript 实现文件上传功能；文本域<textarea>标签是可以输入多行文本的元素控件。

【操作步骤】

第 1 步，新建文档，设计表单结构，代码如下所示，在浏览器中显示效果如图 8.15 所示。

```
    <div class="feedback">
    <h3>反馈表单</h3>
    <div class="content">
        <form method="post" action="">
            <fieldset class="base_info">
                <legend>用户信息</legend>
                <div class="frm_cont userName"><label for="userName">用户名：</label><input type="text"
value="" id="userName" /></div>
                <div class="frm_cont email"><label for="email">电子邮件：</label><input type="text" value="@"
id="email" /></div>
                <div class="frm_cont url"><label for="url">网址：</label><input type="text" value="http://"
id="url" /></div>
            </fieldset>
            <fieldset class="feedback_content">
                <legend>反馈内容</legend>
                <div class="frm_cont up_file">
                    <label for="up_file">相关图片：</label><input type="file" id="up_file" />
                    <p class="tips">本系统只支持上传.jpg、.gif、.png 图片。</p>
                </div>
                <div class="frm_cont msg">
                    <label for="msg">内容：</label><textarea rows="4" cols="40" id="msg"></textarea>
                    <p class="tips">请输入留言内容！</p>
                </div>
```






```
            </fieldset>
            <div class="btns"><button type="submit">提交</button><button type="reset">重置</button></div>
        </form>
    </div>
    </div>
```

第 2 步，使用 CSS 样式定义 HTML 标签的表现，基本原则是从外到内，从泛到细，更重要的是要善于利用 CSS 选择器。

```
    .feedback {/*定义表单整体的宽度和边框样式等*/
        width:398px;
        padding:1px;
        border:1px solid #E8E8E8;
        background-color:#FFFFFF;
    }
    .feedback * {/*定义表单内部的所有元素内补丁、外补丁以及文字的相关样式*/
        margin:0;
        padding:0;
        font:normal 12px/1.5em "宋体", Verdana,Lucida, Arial, Helvetica, sans-serif;
    }
```

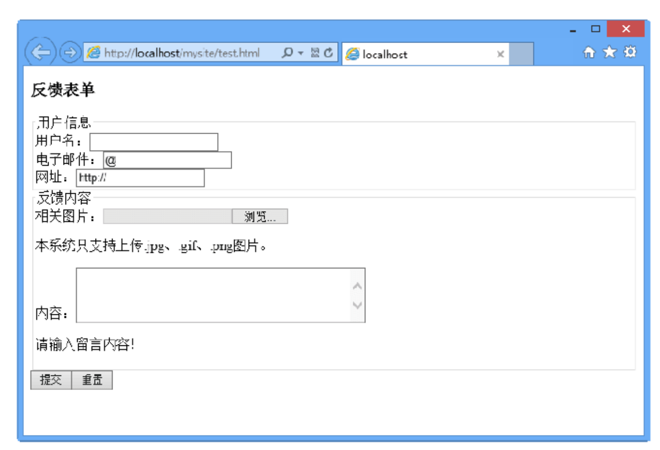


图 8.15　默认解析表单效果

第 3 步，整体样式的定义主要包含反馈表单的整体宽度和内部所有子元素的整体定义。整体宽度、边框等样式的定义是根据视觉效果而设定；定义内部所有子元素的样式是为了提高后期对子元素样式定义的便利性。

```
    .feedback h3 {/*定义表单标题的高度、文字样式以及背景颜色等*/
        height:24px;
        line-height:24px;
        font-weight:bold;
        font-size:13px;
        text-indent:12px;
        color:#FFFFFF;
        background-color:#999999;
    }
```


Note

第 4 步，定义反馈表单标题的高度，并设置标题文本缩进以及文字大小等样式，增强标题与内容之间的反差和整齐感。

```
.feedback .content {/*表单内容区域增加 10px 的左、右内补丁，使其与表单外框产生间距*/
    padding:0 10px;
}
```

第 5 步，为了不让反馈表单内部信息与边框太紧密，将表单内容区域增加 10px 的左、右内补丁，使其与表单整体有一定的间距。

```
.feedback fieldset {/*定义表单域边框样式以及与上下几个元素之间的间距*/
    padding-left:12px; /*因为前面已经将表单内部所有内补丁设置为 0，所以增加 12px 的左内补丁使表单域标题缩进*/
    margin-top:10px;
    border:0 none;                  /*去除默认的表单域边框*/
    border-top:1px solid #999999;   /*定义表单域上边框的样式*/
}
.feedback legend {
    padding:0 5px;                  /*设置表单域的标题在表单域上边框中的间距*/
    color:#333333; /*考虑 IE 浏览器解析表单域标题时文字颜色与 Firefox 浏览器不同，所以统一定义相同的颜色值*/
}
```

表单域在浏览器默认解析的情况下是有边框线的，不需要边框线就需要将其隐藏，需要部分边框线就要将不需要的部分隐藏。只需要一条上边框线，因此首先将所有边框消除（border:0 none;），然后再次定义上边框的样式。

第 6 步，在定义整体样式时，将所有内补丁（padding）定义为 0，导致表单域标题<legend>标签中的文字紧挨表单域边框。需要将其缩进，就要将左内补丁值增加，如“padding-left:12px;”即可将表单域标题缩进，如图 8.16 所示。

```
.feedback .frm_cont {
    margin-top:8px; /*表单内容区域中不同表单之间的上下间距*/
}
```

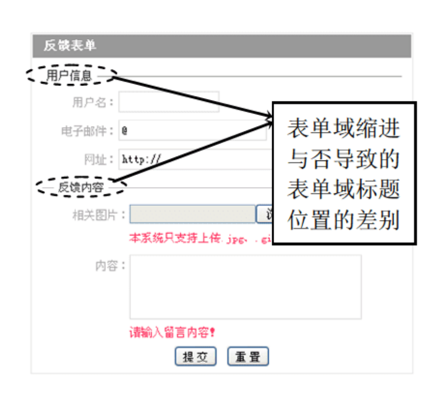


图 8.16　表单域缩进后对表单域标题的影响对比

第 7 步，将每个表单的内容增加外补丁，增加每个表单元素之间的空间感。

```
.feedback label {/*定义 label 标签的宽度、右对齐，设置浮动，使其与输入框并列*/
```






```
        float:left;
        width:80px;
        height:22px;
        line-height:24px;
        text-align:right;
        color:#ABABAB;
        cursor:pointer;
    }
```

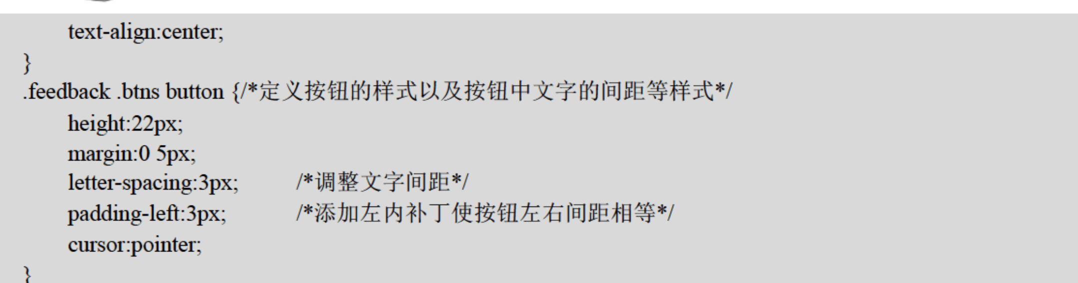
第 8 步，<label>标签使用浮动后可以将旁边的元素（即文本输入框）“吸”到它的旁边，并设置了宽度和高度属性，再将文字右对齐。这样的排列在视觉效果上可以达到整齐的感觉，不会让浏览者感觉这个表单是杂乱无章的。

```
    .feedback .base_info input {/*定义表单内容区域中所有输入框的宽度和高度等样式*/
        width:100px;
        height:17px;
        padding:3px 2px 0;
        border:1px solid #DEDEDE;
    }
    .feedback .email input {/*针对 Email 地址输入框，改变其宽度属性值*/
        width:150px;
    }
    .feedback .url input {/*针对网址输入框，改变其宽度属性值*/
        width:240px;
    }
    /*避免因输入框的高度和宽度修改导致浏览器之间的差别，使用 auto 默认值恢复浏览器默认解析*/
    .feedback .up_file input {
        width:auto;
        height:auto;
    }
```

.feedback .base_info input 选择器为反馈表单中类名为 base_info 的容器内所有的<input>标签设置了宽度、高度和边框样式，再针对不同功能的输入框设置宽度，不仅能加大显示输入数据的空间，还可以形成表单之间有序的错落感。

第 9 步，文件上传控件 input（type="file"）也是<input>标签，但不在类名为 base_info 的容器之内，所以最终显示的还是默认的浏览器解析效果。

```
    .feedback .tips {/*将提示文本利用内补丁缩进，并设置红色，突出显示*/
        padding:5px 0 0 80px;
        color:#FF3260;
    }
    .feedback textarea {/*定义文本域的宽高以及内部文字的行高等样式*/
        width:240px;
        height:66px;
        padding-left:2px;
        line-height:22px;
        border:1px solid #DEDEDE;
    }
    .feedback .btns {/*按钮区域增加上下内补丁，加大间距，并定义其内部的元素居中显示*/
        padding:5px 0;
```



```
    text-align:center;
}
.feedback .btns button {/*定义按钮的样式以及按钮中文字的间距等样式*/
    height:22px;
    margin:0 5px;
    letter-spacing:3px;        /*调整文字间距*/
    padding-left:3px;          /*添加左内补丁使按钮左右间距相等*/
    cursor:pointer;
}
```

Note

第 10 步，将文本域、提示信息和按钮等元素定义为相关样式。文本域中使用“padding-left:2px;”是需要使文字与其边框产生间距；按钮中定义“letter-spacing:3px;”可以让按钮中的文字之间有 3px 的间距，以文字的右边为基准。

第 11 步，经过以上 CSS 修饰的 HTML 结构，最终会在浏览器中显示如图 8.17 所示的页面效果。

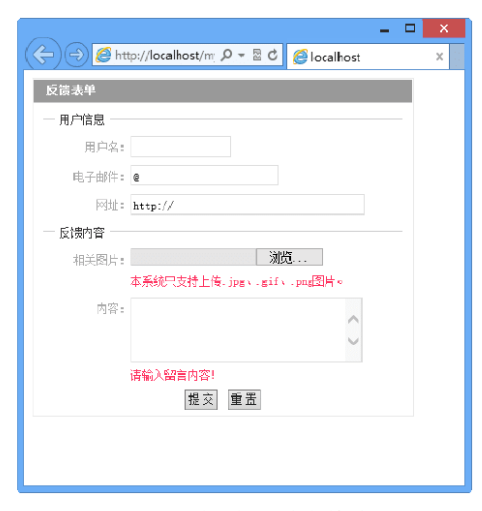


图 8.17　最终的反馈表单效果



视频讲解

8.2.9　设计搜索表单

搜索框一般包含关键词输入框、搜索类别、搜索提示和搜索按钮，当然简单的搜索框只有关键词输入框和搜索按钮这两部分。本案例将介绍如何设计附带提示的搜索框样式，演示效果如图 8.18 所示。



图 8.18　设计搜索框





【操作步骤】

第 1 步，新建一个网页，保存为 index.html，在<body>标签内输入如下结构代码，构建表单结构。

```
<div class="search_box">
    <h3>搜索框</h3>
    <div class="content">
        <form method="post" action="">
            <select>
                <option value="1">网页</option>
                <option value="2">图片</option>
                <option value="3">新闻</option>
                <option value="4">MP3</option>
            </select>
            <input type="text" value="css" /> <button type="submit">搜索</button>
            <div class="search_tips">
                <h4>搜索提示</h4>
                <ul>
                    <li><a href="#">css 视频</a><span>共有 589 个项目</span></li>
                    <li><a href="#">css 教程</a><span>共有 58393 个项目</span></li>
                    <li><a href="#">css+div</a><span>共有 158393 个项目</span></li>
                    <li><a href="#">css 网页设计</a><span>共有 58393 个项目</span></li>
                    <li><a href="#">css 样式</a><span>共有 158393 个项目</span></li>
                </ul>
            </div>
        </form>
    </div>
</div>
```

整个表单结构分为两个部分，将下拉选择框、文本框和按钮归为一类，主要功能是用于搜索信息；搜索提示为当在文本框中输入文字时，将会出现相对应的搜索提示信息，该功能主要由后台程序开发人员实现，前台设计师只需要将其以页面元素表现即可。

第 2 步，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后逐步输入 CSS 代码，设计表单样式。

第 3 步，通过分析最终效果可以看到，页面中并没有显示“站内搜索”和“搜索提示”这两个标题，且搜索按钮是以图片代替的，搜索提示出现在搜索输入框的底部，并且宽度与输入框相同。为此，开始在内部样式表中输入下面样式，对表单结构进行初始化设计。

```
.search_box {/*设置输入框宽度，并设置为相对定位，成为其子级元素的定位参考*/
    position:relative;
    width:360px;
}
.search_box * {/*设置输入框内补丁、边界为 0，列表修饰为无，并且设置字体样式等*/
    margin:0; padding:0;
    list-style:none;
    font:normal 12px/1.5em "宋体", Verdana,Lucida, Arial, Helvetica, sans-serif;
}
.search_box h3, .search_tips h4 {display:none;} /*隐藏标题文字*/
```

第 4 步，设置搜索框整体的宽度属性值以及其所有子元素的内补丁、边界等相关属性。为了方便将搜索提示信息框通过定位的方式显示在搜索输入框的底部，在.search_box 中定义 position 属性，让



其成为子级元素定位的参照物。文档结构中的标题在页面中不需要显示，因此可以将其隐藏。现在只是将标题文字隐藏了，后期网站开发过程如果需要显示时，可以直接通过 CSS 样式修改，而不需要再次调整文档结构。



Note

```
    .search_box select {/*将下拉框设置为浮动，并设置其宽度值*/
        float:left;
        width:60px;
    }
    .search_box input {/*设置搜索输入框样式，浮动显示，添加左右两边间距（边界）*/
        float:left;
        width:196px; height:14px;
        padding:1px 2px; margin:0 5px;
        border:1px solid #619FCF;
    }
    .search_box button {/*设置按钮浮动，以缩进方式隐藏按钮文字，添加背景图*/
        float:left;
        width:59px; height:18px;
        text-indent:-9999px;
        border:0 none;
        background:url(images/btn_search.gif) no-repeat 0 0;
        cursor:pointer;
    }
```

第 5 步，搜索类别下拉框、搜索关键字输入框和搜索按钮这 3 个元素按照常理来理解原本就是可以并列显示的，但为了将这 3 个元素之间的默认空间缩短，使用了“float:left;”，再利用输入框 input 增加可控的边界“margin:0 5px;”调整三者之间的间距。

三者之间整体样式调整完毕后，再对其细节部分进行详细的调整修饰。美化输入框并且利用文字缩进属性隐藏按钮上的文字，使用图片代替。

第 6 步，下拉框<select>标签只是设置了宽度属性值，并未设置其高度属性值，其中的原因就是 IE 浏览器和 Firefox 浏览器对其高度属性值的解析完全不一样，因此采用默认的方式而不是再次利用 CSS 样式定义其相关属性。

第 7 步，按钮<button>标签在默认情况下不具备当鼠标悬停时显示手形指针，因此需要特殊定义。

```
    .search_tips {/*将搜索提示框设置的宽度与输入框相等，并绝对定位在输入框底部*/
        position:absolute;
        top:17px; left:65px;
        width:190px; padding:5px 5px 0;
        border:1px solid #619FCF;
    }
```

第 8 步，搜索提示框使用绝对定位的方式显示在输入框的底部，其宽度属性值等于输入框的宽度属性值，使页面更加美观。不设置提示框的高度属性值是希望搜索框能随着内容的增加而自适应高度。

```
    .search_tips li {/*设置搜索提示框内的列表宽度和高度值，利用浮动避免 IE 浏览器中列表上下间距增多的
BUG*/
        float:left;
        width:100%;
        height:22px;
        line-height:22px;
    }
```



第 9 步，在 IE 早期版本浏览器中，列表<li>标签上下间距会增大显示，为了避免该问题的出现，为所有列表<li>标签添加浮动 float 属性。宽度属性值设置为 100%可以避免当列表<li>标签具有浮动属性时，宽度自适应的问题。

```
.search_tips li a {/*搜索提示中相关文字居左显示，并设置相关样式*/
    float:left;
    text-decoration:none;
    color:#333333;
}
.search_tips li a:hover {/*搜索提示中相关文字在鼠标悬停时显示红色文字*/
    color:#FF0000;
}
.search_tips li span {/*以灰色弱化搜索提示相关数据，并居右显示*/
    float:right;
    color:#CCCCCC;
}
```

第 10 步，将列表项<li>标签中的锚点<a>标签和<span>标签分别左右浮动，使它们靠两边显示在搜索提示框内，并相应地添加文字样式做细节调整。

8.3　在线练习

本节通过大量的上机示例，帮助初学者练习使用 HTML5 设计表单结构和样式。感兴趣的读者可以扫码练习：（1）表单行为；（2）表单美化。



在线练习 1



在线练习 2



第9章 CSS 盒模型

1996 年 W3C 推出 CSS，并规定了页面中所有元素基本显示形态为方形的盒子（Box），并由此形成了一套严谨的盒子模型（Box Model）。根据这个盒模型规则，网页中所有元素对象都被放在一个盒子里，设计师可以通过 CSS 来控制这个盒子的显示属性，这就是经典的 CSS 盒模型。盒模型是 CSS 基础，它规定了网页元素显示方式，以及如何控制元素间的位置关系，本章将围绕盒模型的缘起、概念、结构、尺寸等基础知识展开讲解，为学习和使用 CSS 进行网页设计奠定扎实的基础。

【学习重点】

- 了解 CSSS2 盒模型。
- 设计边框样式。
- 设计边界样式。
- 设计补白样式。





9.1　盒模型基础

CSS 定义所有的元素都可以拥有像盒子一样的外形和平面空间，它包含边界、边框、补白、内容，内容又包括填充对象和背景（背景色和背景图像）。盒模型规范了网页元素的显示基础，关系到网页设计中排版、布局、定位等操作，所有元素都必须遵循盒模型规则。

9.1.1　盒模型概述

传统网页设计是以表格为基础的，所有网页元素都依靠表格来确定自己的外框和位置。但表格布局也给网页带来很多问题。例如，在表格页面中，为了给一段文本加上一个彩色的边框，需要添加一个表格，然后为表格定义边框，通过这种间接的方式为文本增加一个边框。现在想一想，这似乎有点不可思议，但传统布局中这种应用确实很普遍，因此一个网页中可能需要添加很多个类似的无意义的表格。

CSS 完全抛弃了表格的束缚，明确规定了网页中所有元素都可以定义自己的模型。除了边框外，在元素内容四周还可以定义一个空白区域以控制元素边框与元素内容之间的位置关系，以及在边框外边定义一个空白区域以控制元素与其他元素的距离。

元素内容与边框之间的空白区域，被称作元素的补白（padding），有人也称之为元素的内边距、填充或内框；元素边框外边的空白区域，被称作边界（margin），有人也称之为元素的外边距或外框。

由于每个元素都可以拥有自己的模型，它看起来像一个矩形的盒子，于是有人就把 CSS 的这种规则统称为盒模型。

盒模型概念的提出，使网页布局完全摆脱了表格的束缚，任何元素对象，无论是传统的段落、列表、标题、图片，还是标准布局中的 div 和 span 元素，都可以通过自己的属性来实现布局，设置模型显示效果。

CSS3 规范新增加了 UI 模块（User-interface 样式模块），该模块用来控制与用户界面相关效果的呈现方式，详细资料请参阅 http://www.w3.org/TR/css3-ui/。该模块改善了传统盒模型结构，增强了盒子构成要素的功能，扩展了盒模型显示的方式，具体描述如下。

☑　改善结构：除了传统的内容区、边框区、补白区和边界区外，为盒子新增轮廓区。

☑　增强功能：内容区增强 CSS 自动添加内容功能，增强内容溢出、换行处理，此部分知识可参阅第 4 章内容；允许多重定义背景图、控制背景图显示方式等，此部分知识可参阅第 5 章内容；增加背景图边框、多重边框、圆角边框等功能，此部分知识可参阅第 12 章内容；完善“margin:auto;”布局特性，此部分知识可参阅第 11 章内容。

☑　扩展显示：完善传统的块显示特性，增加弹性、伸缩盒显示功能，丰富网页布局手段，此部分知识可参阅第 11 章内容。



9.1.2　盒模型结构

在网页设计中，经常会听到内容（content）、补白（padding）、边框（border）、边界（margin），CSS 盒模式都具备这些属性。用户可以把盒模式转移到日常生活中的盒子（箱子）上来理解，日常生活中所见的盒子也就是能装东西的一种箱子，也具有这些属性，所以叫盒子模型。盒模型具有如下特点，结构示意如图 9.1 所示。





Note

☑ 每个盒子都有边界、边框、补白、内容 4 个属性。
☑ 每个属性都包括 4 个部分：上、右、下、左。
☑ 每个属性可同时设置，也可分别设置。
☑ 边界和补白只能够定义大小，而边框可以定义样式、大小和颜色。
☑ 内容可以定义宽度和高度。

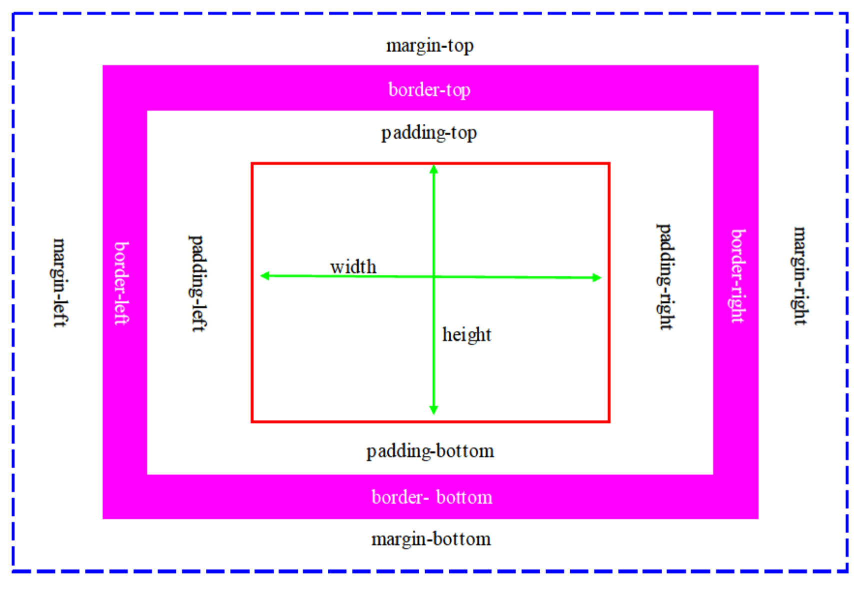


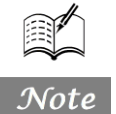
图 9.1　盒模型结构示例图

【示例】CSS 盒模型是网页设计的基础，下面示例在文档中插入一个 div 元素，它遵循盒模型规则，定义 div 元素的 margin、background-color、background-image、padding、border 属性和文本显示内容。在 Dreamweaver 编辑窗口中选中后的预览效果如图 9.2 所示，可以看到 Dreamweaver 能够可视化模拟出盒模型结构，用户能够很清楚地看出盒模型中各个属性的效果。

```
<style type="text/css">
#box {
    height:400px;                              /*定义元素的高*/
    width:400px;                               /*定义元素的宽*/
    margin:60px;                               /*定义元素的边界*/
    padding:60px;                              /*定义元素的补白*/
    border:solid 60px #aaa;                    /*定义元素的边框*/
    background-image:url(images/box1.jpg);     /*定义元素的背景图像*/
    background-repeat:no-repeat;
    background-color:#CC99CC;                  /*定义元素的背景颜色*/
}
</style>
<div id="box">盒模型结构示意图</div>
```

CSS 规定页面中所有元素都包括 4 个区域：内容区、补白区、边框区和边界区。利用 CSS 属性，可以给元素的 4 个区域设置大小。在默认状态下，所有元素盒模型的初始状态：margin、border、padding、width 和 height 都显示为 0，背景为透明。当元素包含内容后，width 和 height 会自动调整为内容的宽度和高度，而当元素浮动时，情况就非常复杂。





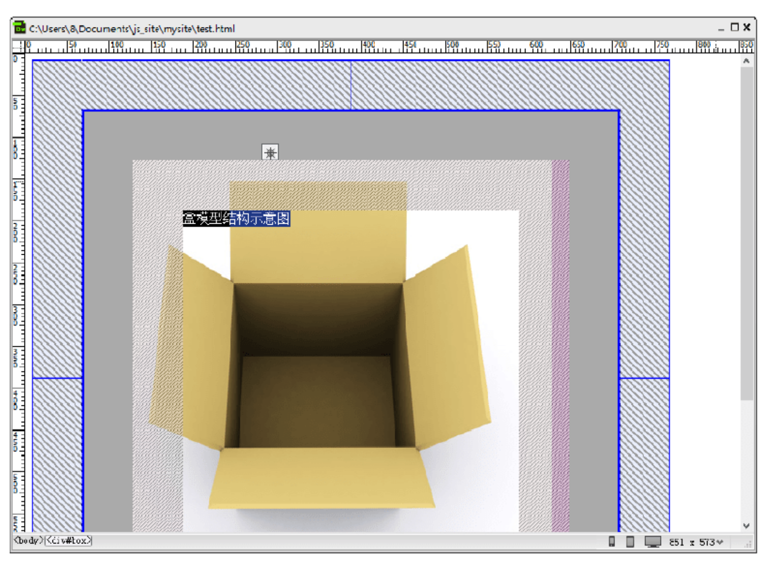


图 9.2　盒模型结构

在 CSS 中，可以增加补白、边框和边界的区域大小，这些不会影响内容区域（宽和高），但会增加元素框的总尺寸。

9.1.3　定义盒模型大小



视频讲解

CSS 盒子模型使用 width（宽）和 height（高）定义内容区域的大小，但是很多用户误以为 width 属性表示整个元素的宽度，height 属性表示整个元素的高度，包括 IE 老版本的浏览器都这样认为。

【示例 1】下面示例定义两个并列显示的 div 元素，设置每个 div 的 width 为 50%，显示效果如图 9.3 所示。

```
<style type="text/css">
div {/*定义 div 元素公共属性*/
    float: left;                              /*向左浮动，实现并列显示*/
    background-image: url(images/1.jpg);      /*定义背景图像*/
    background-color: #CC99CC;                /*定义背景色*/
    font-size: 32px;                          /*定义 div 内显示的字体大小*/
    color: #FF0000;                           /*定义 div 内显示的字体颜色*/
    text-align: center;                       /*定义 div 内显示的字体居中显示*/
    height: 540px;                            /*定义高度*/
}
#box1 {/*定义第 1 个 div 元素属性*/
    width: 50%;                               /*占据窗口一半的宽度*/
}
#box2 {/*定义第 2 个 div 元素属性*/
    width: 50%;                               /*占据窗口一半的宽度*/
}
</style>

<div id="box1">左边元素</div>
<div id="box2">右边元素</div>
```




Note

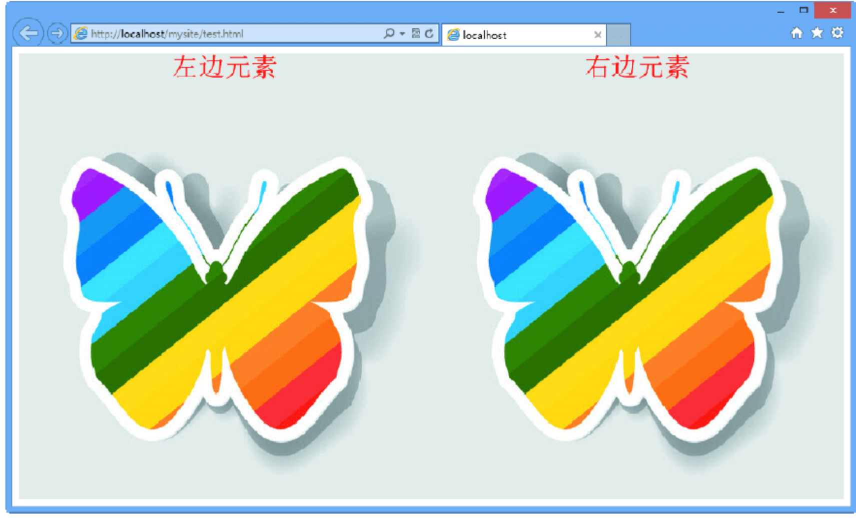


图 9.3　定义元素的大小

【示例 2】 下面示例以示例 1 为基础，设计两个 div 元素并列显示，同时设置元素间留出一点空隙，以便区分不同元素。

```
<style type="text/css">
div {/*定义 div 元素公共属性，具体声明与示例 1 代码相同，就不再重复注释*/
    float:left;
    background-image:url(images/1.jpg);
    background-color:#CC99CC;
    font-size:32px;
    color:#FF0000;
    text-align:center;
    height:540px;}
#box1 {/*定义第 1 个 div 元素属性*/
    width:50%;
    margin-right:1px; /*定义右边界为 1px 宽*/
}
#box2 {/*定义第 2 个 div 元素属性*/
    width:50%;
    margin-left:1px; /*定义左边界为 1px 宽*/
}
</style>

<div id="box1">左边元素</div>
<div id="box2">右边元素</div>
```

在浏览器中预览，可以发现第 2 个 div 元素错行显示，如图 9.4 所示。这是因为当左右两个 div 的 width 属性都设置为 50%时，div 内容本身就会挤满浏览器窗口，由于 margin 属性不包含在 width 属性之内，所以两个 div 就无法同时显示在一行内。

解决方法：可以设置 width 属性为 49%或更小。但是当空隙设置较大时，通过缩减百分比还是存在一定的风险或局限性。此时，最好的解决办法是设置 width 为固定值，不过这样会使页面布局失去灵活性。

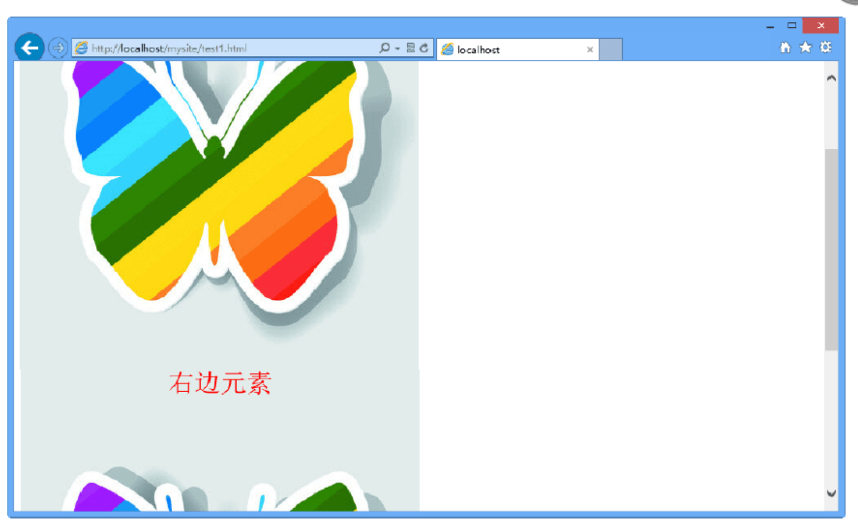
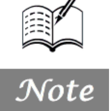


图 9.4　错行显示效果



Note

【拓展】

用户可以使用 3 种方法实现在块元素周围生成空隙。第 1 种是只为元素添加补白，第 2 种是只添加边界，第 3 种是既增加补白，又增加边界。

如果元素定义了背景，就不能使用补白设计间距，因为背景会扩展到补白区域内，从而遮盖空隙，使其无法显示。

根据 CSS 盒模型规则，可以给出一个简单的盒模型尺寸计算公式：

☑　元素的总宽度=左边界 ＋ 左边框 ＋ 左补白 ＋ 宽 ＋ 右补白 ＋ 右边框 ＋ 右边界

☑　元素的总高度=上边界 ＋ 上边框 ＋ 上补白 ＋ 高 ＋ 下补白 ＋ 下边框 ＋ 下边界

例如，假设一个元素的宽度为 200px，左右边界为 50px，左右补白为 50px，边框为 20px。则该元素在页面中实际占据宽度为：

50px + 20px + 50px + 200px + 50px + 20px + 50px = 440px

但由于 IE 早期版本（5.x 及以下版本）浏览器对于盒模型的解释使用一种非标准规则，它认为元素的宽度应为内容宽度、补白宽度和边界宽度的总和，用公式表示为：

width=border-left + padding-left + content-width + padding-right + border-fight

height= border-top + padding-top + content-height + padding-bottom + border-bottom

因此，在 IE 早期版本（5.x 及以下版本）浏览器中，元素在页面中所占据的实际大小为：

☑　元素的总宽度=左边界 ＋ 宽 ＋ 右边界

☑　元素的总高度=上边界 ＋ 高 ＋ 下边界

因此，在网页布局时，遇到与宽度或高度相关问题，设计师一定要对盒模型中的 margin、padding 和 border 等属性进行综合考虑，只有这样才能设计出满意的布局，避免出现错位、错行等现象。

9.2　边　　框

在网页中，很多修饰性线条都是由边框定义的，边框可以作修饰使用，也可以作为版块、对象分隔使用，元素的边框具有下面 4 个特点：

☑　每个元素都包含 4 个方位的边框，如 border-top（顶边）、border-right（右边）、border-bottom




（底边）和 border-left（左边），可以单独定义，也可以使用 border 属性统一定义边框样式。

☑ 独立定义边框的宽度，如 border-width、border-top-width、border-right-width、border-bottom-width 和 border-left-width。也可以使用 border-width 属性统一定义边框的宽度。

☑ 独立定义边框的颜色，如 border-color、border-top-color、border-right-color、border-bottom-color 和 border-left-color。也可以使用 border-color 属性统一定义边框的颜色。

☑ 独立定义边框的样式，如 border-style、border-top-style、border-right-style、border-bottom-style 和 border-left-style。也可以使用 border-style 属性统一定义边框的样式。

Note

视频讲解

9.2.1 定义宽度

定义边框的宽度有多种方法，简单说明如下。

方法 1，直接在属性后面指定宽度值。

```
<style type="text/css">
border-bottom-width:12px;          /*定义元素的底边框宽度为 12px*/
border-top-width:0.2em;            /*定义顶部边框宽度为元素内字体大小的 0.2 倍*/
</style>
```

方法 2，使用关键字，如 thin、medium 和 thick。thick 比 medium 宽，而 medium 比 thin 宽。不同浏览器对此解析的宽度值也不同，有的解析为 5px、3px、2px，有的解析为 3px、2px、1px。

方法 3，单独为元素的某条边设置宽度，可以使用 border-top-width（顶边框宽度）、border-right-width（右边框宽度）、border-bottom-width（底边框宽度）和 border-left-width（左边框宽度）。

方法 4，使用 border-width 属性快速定义边框宽度，例如：

```
<style type="text/css">
border-width:2px;                  /*定义四边都为 2px*/
border-width:2px 4px;              /*定义上下边为 2px，左右边为 4px*/
border-width:2px 4px 6px;          /*定义上边为 2px，左右边为 4px，底边为 6px*/
border-width:2px 4px 6px 8px;      /*定义上边为 2px，右边为 4px，底边为 6px，左边为 8px*/
</style>
```



视频讲解

提示： 当定义边框宽度时，必须要定义边框的显示样式。由于边框默认样式为 none，即不显示，所以仅设置边框的宽度，由于样式不存在，边框宽度也自动被清除为 0。

9.2.2 定义颜色

定义边框颜色可以使用颜色名、RGB 颜色值或十六进制颜色值。

【示例】 下面示例分别为元素的各个边框定义不同的颜色，演示效果如图 9.5 所示。

```
<style type="text/css">
#box {/*定义边框的颜色*/
    height: 164px;                 /*定义盒的高度*/
    width: 240px;                  /*定义盒的宽度*/
    padding: 2px;                  /*定义内补白*/
    font-size: 16px;               /*定义字体大小*/
    color: #FF0000;                /*定义字体显示颜色*/
    border-style: solid;           /*定义边框为实线显示*/
    border-width: 50px;            /*定义边框的宽度*/
```



```
    border-top-color: #aaa;                 /*定义顶边框颜色为十六进制值*/
    border-right-color: gray;               /*定义右边框颜色为名称值*/
    border-bottom-color: rgb(120,50,20);    /*定义底边框颜色为 RGB 值*/
    border-left-color:auto;                 /*定义左边框颜色继承字体颜色*/
}
</style>

<div id="box"><img src="images/1.jpg" width="240" height="164" alt=""/></div>
```



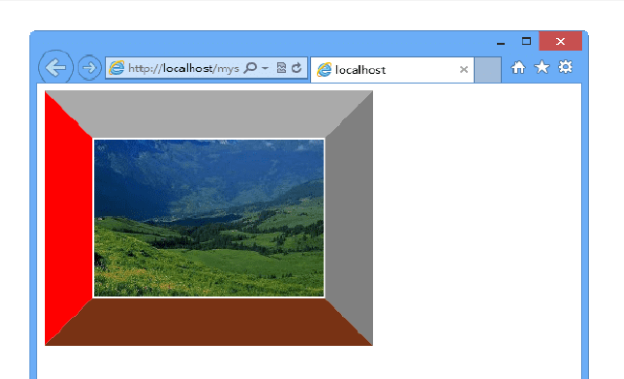


图 9.5　定义边框颜色

9.2.3　定义样式



视 频 讲 解

边框样式是边框显示的基础，CSS3 提供了下面几种边框样式。

- ☑ none：默认值，无边框，不受任何指定的 border-width 值影响。
- ☑ hidden：隐藏边框，IE 不支持。
- ☑ dotted：定义边框为点线。
- ☑ dashed：定义边框为虚线。
- ☑ solid：定义边框为实线。
- ☑ double：定义边框为双线边框，两条线及其间隔宽度之和等于指定的 border-width 值。
- ☑ groove：根据 border-color 值定义 3D 凹槽。
- ☑ ridge：根据 border-color 值定义 3D 凸槽。
- ☑ inset：根据 border-color 值定义 3D 凹边。
- ☑ outset：根据 border-color 值定义 3D 凸边。

【示例 1】下面示例使用边框样式设计列表框样式，定义每个项目显示下画线，预览效果如图 9.6 所示。

```
<style type="text/css">
#box {/*<定义信纸的外框>*/
    width: 500px;
    height: 400px;
    padding: 8px 24px;
    margin: 6px;
    border-style: outset;                   /*定义信纸边框为 3D 凸边效果*/
    border-width: 4px;                      /*定义信纸边框宽度*/
    border-color: #aaa;                     /*定义信纸边框颜色*/
```

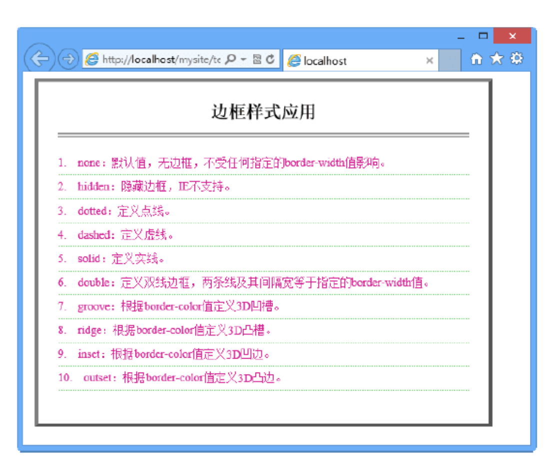




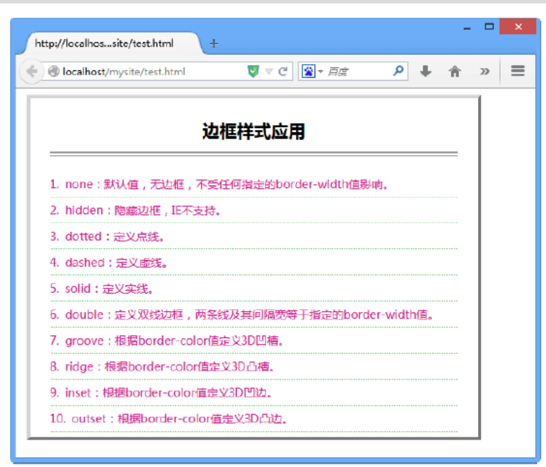
Note

```
        font-size: 14px;
        color: #D02090;
        list-style-position: inside;            /*定义列表符号在内部显示*/
}
#box h2 {/*<定义标题格式>*/
        padding-bottom: 12px;
        border-bottom-style: double;            /*定义标题底边框为双线显示*/
        border-bottom-width: 6px;               /*定义标题底边框宽度*/
        border-bottom-color: #999;              /*定义标题底边框颜色*/
        text-align: center;
        color: #000000;}
#box li {
        padding: 6px 0;                         /*增加列表项之间的间距*/
        border-bottom-style: dotted;            /*定义列表项底边框为点线显示*/
        border-bottom-width: 1px;               /*定义列表项底边框宽度*/
        border-bottom-color: #66CC66;           /*定义列表项底边框颜色*/
}
</style>

<ol id="box">
        <h2>边框样式应用</h2>
        <li>none：默认值，无边框，不受任何指定的 border-width 值影响。</li>
        <li>hidden：隐藏边框，IE 不支持。</li>
        <li>dotted：定义点线。</li>
        <li>dashed：定义虚线。</li>
        <li>solid：定义实线。</li>
        <li>double：定义双线边框，两条线及其间隔宽等于指定的 border-width 值。</li>
        <li>groove：根据 border-color 值定义 3D 凹槽。</li>
        <li>ridge：根据 border-color 值定义 3D 凸槽。</li>
        <li>inset：根据 border-color 值定义 3D 凹边。</li>
        <li>outset：根据 border-color 值定义 3D 凸边。</li>
</ol>
```



IE 预览效果



Firefox 预览效果

图 9.6　边框样式比较



在 IE 和 Firefox 浏览器中分别进行预览，则效果存在细微区别，说明不同浏览器在解析相同的样式代码时显示效果也不完全相同。



提示：当同时定义边框样式、宽度和颜色时，分别输入代码有点烦琐，这时可以合并样式，且属性值顺序可以任意排列：

```
/*[边框样式代码简写]*/
#box { border:outset 4px #aaa;                /*定义信纸边框样式*/
}
#box h2 {border-bottom: 6px #999 double;       /*定义标题底边框样式*/
}
#box li {border-bottom: #66CC66 dotted 1px;    /*定义列表项底边框样式*/
}
```

【拓展】

元素的背景与边框有着某种特殊的关系。根据 CSS 规则，元素的背景不会超出边框的外边缘，由于平时定义边框宽度多为 1px 或 2px，且多为实线，所以察觉不到背景深入到边框的外边缘。不过目前浏览器对此解释还存在争议。

【示例 2】下面示例为盒子设计背景图像，然后设计边框为虚线显示，同时定义边框宽度为 50px，以方便观看。

```
<style type="text/css">
#box {/*查看边框与背景图像的关系*/
    height: 200px;                          /*定义盒子的高度*/
    width: 200px;                           /*定义盒子的宽度*/
    background-image: url(images/2.jpg);    /*定义盒子的背景图像*/
    border: 50px dotted red;                /*定义边框样式为间隔显示*/
}
</style>

<div id="box"></div>
```

在 IE 怪异模式下浏览，则背景没有伸入边框区域，且解析 dotted 样式为圆形，如图 9.7 所示。在 Firefox 中浏览，则背景伸入边框区域，且解析边框虚线样式也不同，如图 9.8 所示。

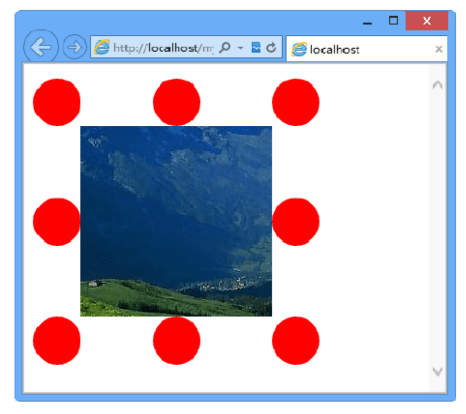


图 9.7　IE 解析边框虚线

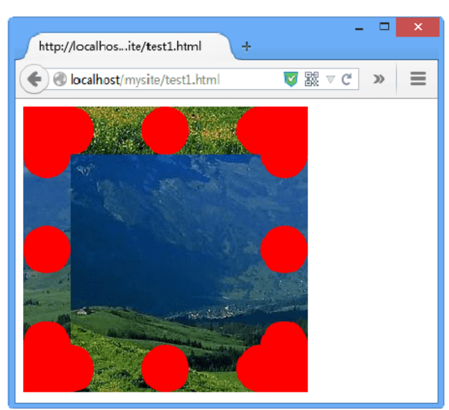


图 9.8　Firefox 解析边框虚线



> 提示：在默认状态下，边框的宽度为 medium（中型），这是一个相对宽度，一般为 2～3px。边框默认样式为 none，即隐藏边框显示。边框默认颜色为前景色，即元素中包含文本的颜色，如果没有文字，则将继承上级元素所包含的文本颜色。



Note



视频讲解

9.2.4 案例：设计行内元素边框

根据盒模型基本规则，任何元素都可以定义边框，但行内元素的边框显示效果有点特殊。下面结合示例进行简单说明。

第一，行内元素的上下边框高度不会影响行高，而且不受段落和行高的约束。

【示例 1】下面示例在一段文本中包含一个 span 元素，利用它为部分文本定义特殊样式，设计顶部边框为 80px 的红色实线，底部边框为 80px 的绿色实线，如图 9.9 所示。

```
<style type="text/css">
p {/*定义段落属性*/
    margin: 50px;                        /*定义段落的边界为 50px*/
    border: dashed 1px #999;             /*定义段落的边框*/
    font-size: 14px;                     /*定义段落字体大小*/
    line-height: 24px;                   /*定义段落行高为 24px*/
}
span {/*定义段落内内联文本属性*/
    border-top: solid red 80px;          /*定义行内元素的上边框样式*/
    border-bottom: solid green 80px;     /*定义行内元素的下边框样式*/
    color: blue;
}
</style>

<p> 寒蝉凄切，对长亭晚，骤雨初歇。都门帐饮无绪，留恋处，兰舟催发。执手相看泪眼，竟无语凝噎。念去去，千里烟波，暮霭沉沉楚天阔。 <span>多情自古伤离别，更那堪，冷落清秋节！</span>今宵酒醒何处?杨柳岸，晓风残月。此去经年，应是良辰好景虚设。便纵有千种风情，更与何人说? </p>
```

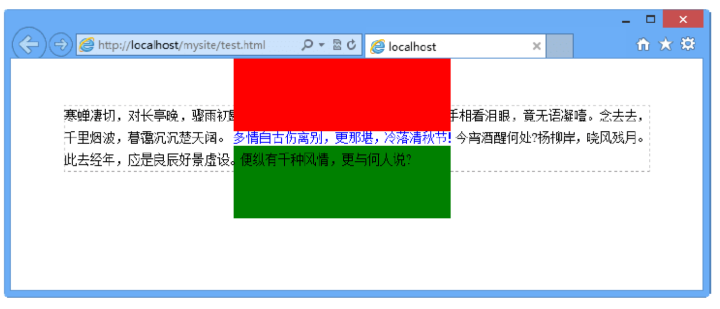


图 9.9 定义行内元素上下边框效果

在 IE 中浏览，可以看到上边框压住了上一行文字，并超出了段落边框，下边框压住了下一行文字，也超出了段落边框。

第二，行内元素的左右边框宽度会挤占左右相邻文本的位置，而不是压住左右两侧文本。左右边框会跟随文本流自由移动，移动时会紧跟行内元素前后，且不会出现断行现象，也就是说单个边框不会被分开显示在两行内。

【示例 2】下面示例在一段文本中包含一个 span 元素，利用它为部分文本定义特殊样式，设计左



侧边框为 60px 的红色实线，右侧边框为 20px 的蓝色实线，上下边框为 1px 的红色实线。在 IE 中浏览，左右边框分别占据一定的位置，效果如图 9.10 所示。

```
<style type="text/css">
p {/*定义段落属性*/
    margin:20px;
    border:dashed 1px #999;
    font-size:14px;
    line-height:24px;}
span {/*定义段落内内联文本属性*/
    border-left:solid red 60px;         /*定义行内元素的左边框样式*/
    border-right:solid blue 20px;       /*定义行内元素的右边框样式*/
    border-top:solid red 1px;           /*定义行内元素的上边框样式*/
    border-bottom:solid red 1px;        /*定义行内元素的下边框样式*/
    color:#aaa;                         /*定义字体颜色*/
}
</style>

<p> 寒蝉凄切，对长亭晚，骤雨初歇。都门帐饮无绪，留恋处，兰舟催发。执手相看泪眼，竟无语凝噎。念去去，千里烟波，暮霭沉沉楚天阔。 <span>多情自古伤离别，更那堪，冷落清秋节！</span>今宵酒醒何处?杨柳岸，晓风残月。此去经年，应是良辰好景虚设。便纵有千种风情，更与何人说? </p>
```

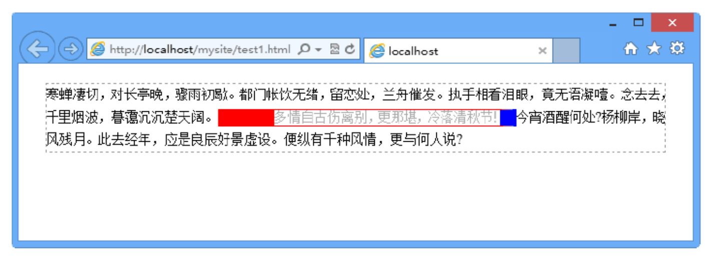


图 9.10　定义行内元素左右边框效果

行内元素的盒模型比较特殊，它的形状也失去了基本矩形形状，有时由于多行显示会呈现为非矩形形状，在设计中需要用户留心这些差异。

9.3　边　界

元素的外部空隙被称为盒模型的边界，也就是元素与元素之间的间距。边界是网页布局中另一个要素，恰当设置边界可以使网页布局疏朗有致，整体看起来优美得体。

9.3.1　定义边界



设置边界可以使用 margin 属性，例如：

```
margin:2px;                 /*定义元素四边边界为 2px*/
margin:2px 4px;             /*定义上下边界为 2px，左右边界为 4px*/
margin:2px 4px 6px;         /*定义上边界为 2px，左右边界为 4px，下边界为 6px*/
margin:2px 4px 6px 8px;     /*定义上边界为 2px，右边界为 4px，下边界为 6px，左边界为 8px*/
```



也可以使用 margin-top、margin-right、margin-bottom、margin-left 属性独立设置上、右、下和左边界的大小，例如：



```
margin-top:2px;                /*定义元素上边界为 2px*/
margin-right:2em;              /*定义右边界为元素字体的 2 倍*/
margin-bottom:2%;              /*定义下边界为父元素宽度的 2%*/
margin-left:auto;              /*定义左边界为自动*/
```

margin 可以使用任何长度单位，如像素、磅、英寸、厘米、em、百分比等。margin 默认值为 0，如果没有定义 margin 的值，则意味着元素没有边界。

在实际使用中，各种浏览器都会为一些元素预定义边界样式，如 p、h1～6、ul 等，在显示时浏览器也会自动显示一定的边界。因此一些元素虽然没有定义 margin，但并不意味它们没有边界。



视频讲解

9.3.2 案例：边界的应用

1．网页居中

auto 是一个自动计算的值，该值一般为 0，也可以为其他值，这主要由具体浏览器来确定。

【示例 1】auto 有一个重要作用就是用来实现元素居中显示，下面示例演示了如何设计页面居中显示，效果如图 9.11 所示（test.html）。

```
<style type="text/css">
body { text-align:center; }            /*在 IE 浏览器下实现居中显示*/
div#page {
    margin:5px auto;                   /*在非 IE 浏览器下实现居中显示*/
    width:910px;
    height:363px;
    background-image:url(images/1.png);
    border:solid red 1px;}
</style>

<div id="page">模拟页面</div>
```



图 9.11　居中显示效果

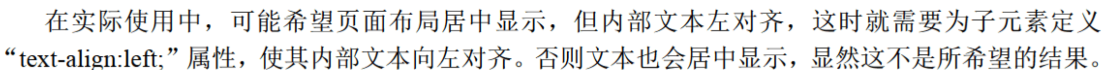


要实现 CSS 平行居中，首先应在父元素中定义“text-align:center;”，这个规则在 IE 早期版本浏览器中可以实现父元素内的所有内容，包括文本、行内元素和块状元素居中显示。但在其他浏览器中只能实现文本、行内元素居中显示。而要在标准浏览器中实现块状元素居中显示，解决方法是为显示元素定义“margin-right:auto;margin-left:auto;”属性。



如果想用这种方法使整个页面居中，建议不要把所有模块都套在一个 div 元素里，可以根据上面示例 CSS 布局代码定义，然后为每个模块的包含框元素 div 定义“margin-right:auto;margin-left:auto;”就可以实现该元素居中显示。

在实际使用中，可能希望页面布局居中显示，但内部文本左对齐，这时就需要为子元素定义“text-align:left;”属性，使其内部文本向左对齐。否则文本也会居中显示，显然这不是所希望的结果。



2．设计弹性页面

边界可以设置为百分比，百分比的取值是根据父元素宽度来计算的。使用百分比能够使页面自适应窗口大小，并能够及时调整边界宽度。从这点考虑，选用百分比具有更大灵活性和更多使用技巧。但是，如果父元素的宽度发生变化，则边界宽度也会随之变化，整个版面可能会混乱，因此在综合布局时要慎重选择。不过在结构单纯、内容单一的布局中，适当使用百分比会使页面更具人性化和多变效果。

【示例 2】下面示例通过 margin 取值百分比定义弹性布局页面，效果如图 9.12 所示（test1.html）。

```
<style type="text/css">
#box {/*定义文本框属性*/
    margin:2%;                /*边界为 body 宽度的 2%*/
    padding:2%;               /*补白为 body 宽度的 2%*/
    background:#CCCC33;}
#box #content {               /*定义文本框内文本段的属性*/
    margin:4%;                /*边界为文本框宽度的 4%*/
    line-height:1.8em;        /*定义行高为字体高度的 1.8 倍*/
    font-size:12px;           /*定义字体大小*/
    color:#003333;            /*定义字体颜色*/
}
#box .center {/*居中加粗文本*/
    margin:4%;                /*边界为文本框宽度的 4%*/
    text-align:center;        /*文本居中显示*/
    font-weight:bold;         /*定义标题为粗体*/
}
</style>

<div id="box">
    <p class="center">将进酒</p>
    <p class="center">李白</p>
    <p id="content"> 君不见，黄河之水天上来，奔流到海不复回。<br />
        君不见，高堂明镜悲白发，朝如青丝暮成雪。<br />
        人生得意须尽欢，莫使金樽空对月！<br />
        天生我材必有用，千金散尽还复来。<br />
        烹羊宰牛且为乐，会须一饮三百杯！<br />
        岑夫子，丹丘生，将进酒，君莫停！<br />
        与君歌一曲，请君为我侧耳听！<br />
        钟鼓馔玉不足贵，但愿长醉不愿醒！<br />
        古来圣贤皆寂寞，惟有饮者留其名！<br />
        陈王昔时宴平乐，斗酒十千恣欢谑。<br />
```



```
        主人何为言少钱？径须沽取对君酌。<br />
        五花马，千金裘，呼儿将出换美酒，与尔同销万古愁！</p>
</div>
```





Note

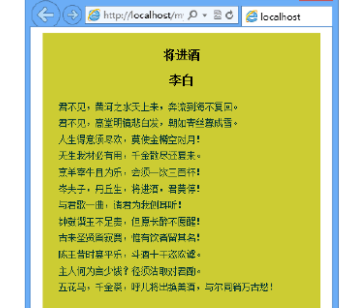




图 9.12　弹性布局效果

在上面示例中，把所有边界都设置为百分比，这样当窗口发生变化时，显示内容也比较得体地成比例变化，不至于当窗口很小时，段落文本所占区域比例很大，当窗口很大时，段落文本所占区域比例又显小气。边界的随机应变使页面更加灵活。

3．调整栏目显示顺序

边界可以取负值，负值边界会给设计带来更多创意，在网页布局中经常应用该技巧。

【示例 3】下面示例模拟一个页面栏目，该栏目包括左右两个分栏，显示效果如图 9.13 所示（test2.html）。

```
<style type="text/css">
#wrap {/*设置栏目包含框样式*/
    width: 997px;                          /*固定栏目总宽度*/
    margin: 12px auto;                     /*定义栏目居中显示*/
}
#box1, #box2 {/*设置左右模块共同属性*/
    float: left;                           /*向左浮动*/
    height: 376px;                         /*固定高度*/
    background-position: center center;    /*背景居中*/
    background-repeat: no-repeat;}         /*背景禁止平铺*/
#box1 {/*定义左侧模块*/
    width: 408px;                          /*固定宽度*/
    background-image: url(images/22.png);} /*定义模拟子栏目效果图*/
#box2 {/*定义右侧模块*/
    width: 589px;                          /*固定宽度*/
    background-image: url(images/23.png);} /*定义模拟子栏目效果图*/
</style>

<div id="wrap">
    <div id="top"><img src="images/21.png" /></div>
    <div id="box1"></div>
    <div id="box2"></div>
</div>
```





图 9.13　默认布局效果



这是一个很普通的两栏布局示意图，如果想把左右两栏位置换一下，似乎很简单，只需要把 HTML 结构调整一下即可：

```
<div id="wrap">
    <div id="top"><img src="images/21.png" /></div>
    <div id="box2"></div>
    <div id="box1"></div>
</div>>
```

但是，当页面很复杂时，各种标签相互嵌套，代码成百上千行，这个看似简单的位置调换，可能牵一发动全身，麻烦不说，甚至破坏布局。

其实，只需要在 CSS 样式表中添加如下两个样式即可，演示效果如图 9.14 所示（test3.html）。

```
#box1 {
    margin-left:589px; /*左栏左边界取正值，值为右栏总宽度的和*/
}
#box2 {
    margin-left:-997px; /*右栏左边界取负值，值为左右栏总宽度的和*/
}
```



图 9.14　百分比取负值布局效果



在浮动布局时，当窗口缩小到一定宽度，如小于或等于左右模块宽度总和时，右边模块就会错行，通过边界取负值能够很好地解决这个问题，且各种浏览器都能够支持。

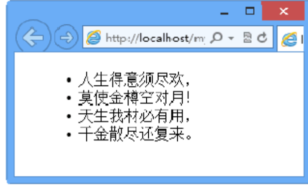


Note

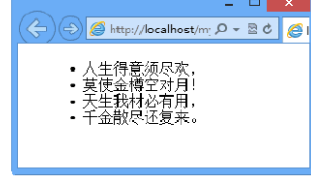
【示例 4】可以使用边界取负值来对段落文本的行距进行一些补偿和修整，下面示例通过 margin 取负值来调整列表项目之间的行距，比较效果如图 9.15 所示（test4.html）。

```
<style type="text/css">
ul {
    margin: 20px;
    font-size: 16px;}
li { margin-top: -2px;                /*压缩列表项之间的空隙*/
}
</style>

<ul>
    <li>人生得意须尽欢，</li>
    <li>莫使金樽空对月！</li>
    <li>天生我材必有用，</li>
    <li>千金散尽还复来。</li>
</ul>
```



压缩前



压缩后

图 9.15　压缩前后比较效果

负边界对文本编排有影响，会间接缩短行距，影响段落的显示效果。另外，可以通过边界与补白的取负配合实现栏目背景色自动向下延伸，利用边界取负实现动态导航效果，通过边界取负隐藏不需要的内容等。



视频讲解

9.3.3　案例：边界重叠现象

在网页排版中，通过 margin 调整栏目之间、对象之间的间距，但是元素之间的 margin 值会发生重叠，影响布局效果，使用时应该小心。简单概括如下：

☑　边界重叠只发生在块状元素，且只是垂直相邻边界才会发生重叠。

☑　边界重叠时，两个边界中最小的那边将被覆盖。

☑　重叠只应用于边界，而补白和边框不会出现重叠。

边界重叠问题由于受各种结构关系的干扰，并非总是按预想的那样显示效果，下面结合示例分别进行讲解。

1．上边元素不浮动，下边元素浮动

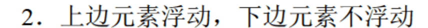
【示例 1】当上边元素不浮动，下边元素浮动时，上下元素的 margin 不会发生重叠（test.html）。

```
<style type="text/css">
/*[边界重叠 1：上边元素不浮动，下边元素浮动]*/
```



```
body {/*清除页边距*/
    margin: 0;                  /*适用 IE*/
    padding: 0;                 /*适用非 IE*/
}
div {/*设置上下元素共同属性*/
    width: 100px;
    height: 100px;
    clear: both;                /*清除并列浮动显示*/
    margin: 20px;
    padding: 20px;}
#box1 {/*定义上边元素不浮动*/ border: solid 20px red;}
#box2 {/*定义下边元素浮动*/ float: left; border: solid 20px blue;}
</style>

<div id="box1">上边元素</div>
<div id="box2">下边元素</div>
```



2．上边元素浮动，下边元素不浮动

【示例 2】与示例 1 相比，这是一个相反的布局，但渲染的效果却截然不同，如图 9.16 所示（test1.html）。

```
<style type="text/css">
body {/*清除页边距*/
    margin:0; /*适用 IE*/
    padding:0; /*适用非 IE*/
}
div {/*定义上下元素共同属性*/
    width:100px;
    height:100px;
    clear:both;
    margin:20px;
    padding:20px;}
#box1 {/*定义上边元素浮动*/
    float:left;
    border:solid 20px red;}
#box2 {/*定义下边元素不浮动*/border:solid 20px blue;}
</style>

<div id="box1">上边元素</div>
<div id="box2">下边元素</div>
```

在 IE 中浏览，上下元素边界发生重叠现象。这与上边元素不浮动，下边元素浮动所讨论的结论是不同的，在其他浏览器中也具有相同的显示效果。

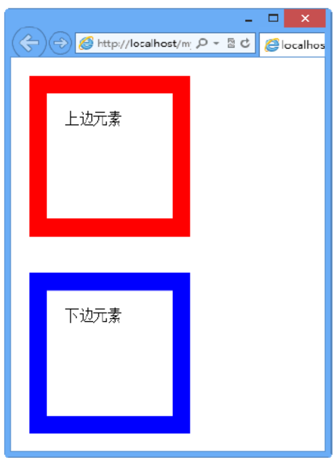
3．一个元素包含另一个元素

一个元素包含另一个元素从结构上讲属于嵌入或包含关系，它们不属于同一级别的相邻关系，外边的元素可以称为父元素，里面的元素可以称为子元素。

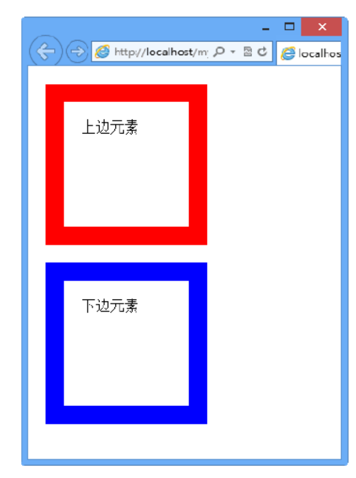








未重叠效果



重叠效果

图 9.16　margin 重叠比较效果

【示例 3】下面示例设计当定义了父元素的边框或补白后，会看到子元素自动停靠在父元素内容框的左上角，如图 9.17 所示（test2.html）。如果内容框内包含其他对象，则自动向下分布。子元素的边界、边框和补白都被包含在内容框里。不管父、子元素是否或独立浮动显示，这种效果在不同浏览器中显示是相同的。

```
<style type="text/css">
body {margin: 0; /*适用 IE*/padding: 0; /*适用非 IE*/
}/*清除页边距*/
div {/*定义父子元素共同属性*/
    margin: 20px;
    padding: 20px;
    float: left;}
#box1 {/*定义父元素的属性*/
    width: 500px;
    height: 300px;
    float: left;
    background-image: url(images/1.jpg);
    border: solid 20px red;}
#box2 {/*定义子元素的属性*/
    width: 150px;
    height: 150px;
    float: left;
    background-image: url(images/2.jpg);
    border: solid 20px blue;}
</style>

<div id="box1">
    <div id="box2">子元素</div>
</div>
```

【示例 4】如果取消父元素的边框和补白，并取消浮动显示，这时会发现子元素的边界越过父元素的边界，实现了叠加，其中较大边界会覆盖较小边界，如图 9.18 所示（test3.html）。



```
<style type="text/css">
body {margin:0; /*适用 IE*/ padding:0; /*适用非 IE*/
}/*清除页边距*/
#box1{ /*不定义父元素的边框、补白，仅设置边界*/
    margin:20px;                        /*父元素的边界为 20px*/
    background-image:url(images/1.jpg);}
#box2 { /*定义子元素的属性*/
    margin:60px;                        /*子元素的边界为 60px*/
    border:solid 20px red;
    padding:20px;
    width:150px;
    height:150px;
    background-image:url(images/2.jpg);}
</style>

<div id="box1">
    <div id="box2">子元素</div>
</div>
```

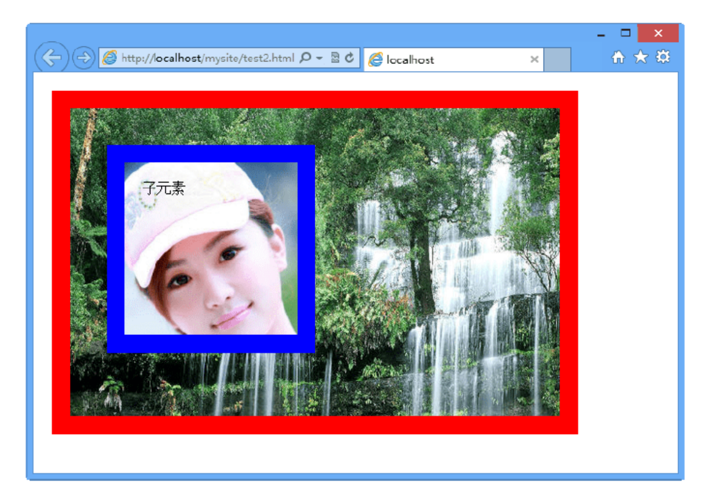


图 9.17　IE 下演示效果

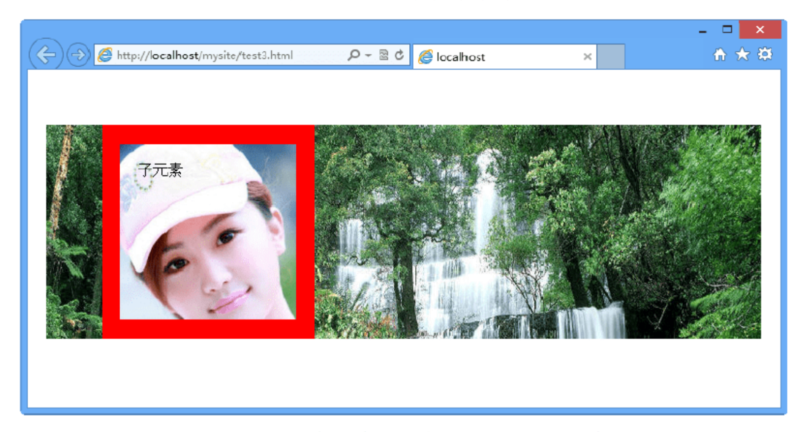


图 9.18　包含元素之间的 margin 重叠现象



提示：边界重叠是网页布局中经常遇到的问题，也是布局中要重点考虑的因素。边界重叠多发生在流动显示的块状元素的上下边界之间，但是如果对象是行内元素、绝对定位的元素，它们的边界一般不会重叠。浮动元素一般也不会发生重叠现象，但它的情况比较复杂，需要结合具体情况进行处理。

在布局中，最好设置上下元素的显示性质相同，例如，如果浮动，则都浮动；如果不浮动，则都不浮动。

如果实际需要，上下元素的显示属性（display）不相同或者必须设置不同的显示性质，如上边元素浮动，下边元素不浮动，或者相反，此时要保证设置浮动元素的清除属性，即“clear:both;”。也可以单独设置下边元素的清除属性，但不能只设置上边元素为左右清除，否则对下边元素无效，因为该属性作用对象为后面解析的元素。

当上下元素显示性质不同时，请务必计算好边界问题，保证版面布局的统一协调。由于不同浏览器对此解释不一，不妨通过 Hack 技术来解决，即通过为不同浏览器设置不同的边界值来实现显示的统一。

Note



视频讲解

9.3.4 行内元素边界

根据盒模型基本规则，任何元素都应有边界。关于行内元素的边界问题比较简单，它没有块状元素那样复杂多变，而且各种浏览器对行内元素的解析效果空前的统一。

与边框一样，行内元素的边界不会改变行高，行高只能由 line-height、font-size 和 vertical-align 属性来改变。与边框一样，行内元素的边界会挤占左右相邻文本的位置，因此使用边界可以调整相邻元素的距离，实现空格。另外，左右边界不会产生断行，边界被浏览器看作一个整体嵌入行内元素的两端。

【示例】下面示例演示了当行内元素定义 margin 之后，它会对左右两侧的间距产生影响，如图 9.19 所示。

```
<style type="text/css">
p {/*影响行高的属性*/
    line-height: 28px;
    font-size: 16px;
    vertical-align: middle;}
span {/*行内元素的边界*/
    margin: 100px;
    border: solid 1px blue;
    color: red;}
</style>

<p> 五月草长莺飞，窗外的春天盛大而暧昧。这样的春日，适合捧一本丰沛的大书在阳光下闲览。<span>季羡林的《清塘荷韵》</span>，正是手边一种：清淡的素色封面，一株水墨荷花迎风而立，书内夹有同样的书签，季羡林的题款颇有古荷风姿。</p>
```

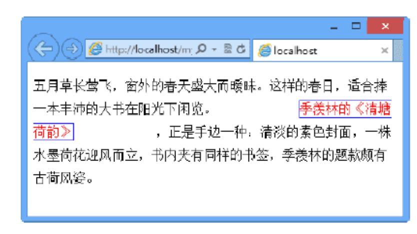


图 9.19　IE 下预览效果





9.4 补　　白

元素的内部空隙被称为盒模型的补白。补白位于元素边框与内容之间，更像 Word 字处理软件中的页边距。

设置补白可以使用 padding 属性，例如：

```
padding:2px;                    /*定义元素四周补白为 2px*/
padding:2px 4px;                /*定义上下补白为 2px，左右补白为 4px*/
padding:2px 4px 6px;            /*定义上补白为 2px，左右补白为 4px，下补白为 6px*/
padding:2px 4px 6px 8px;        /*定义上补白为 2px，右补白为 4px，下补白为 6px，左补白为 8px*/
```

也可以使用 margin-top、margin-right、margin-bottom、margin-left 属性独立设置上、右、下和左边界的大小，例如：

```
padding-top:2px;                /*定义元素上补白为 2px*/
padding-right:2em;              /*定义右补白为元素字体的 2 倍*/
padding-bottom:2%;              /*定义下补白为父元素宽度的 2%*/
padding-left:auto;              /*定义左补白为自动*/
```

与边界不同，补白取值不可以为负。补白和边界一样都是透明的，当设置元素的背景色或边框后，才能感觉到补白的存在。

【示例】下面示例设计导航列表项目并列显示，然后通过补白调整列表项目的显示大小，效果如图 9.20 所示。

```
<style type="text/css">
ul {/*清除列表样式*/
    margin: 0;                  /*清除 IE 列表缩进*/
    padding: 0;                 /*清除非 IE 列表缩进*/
    list-style-type: none;      /*清除列表样式*/
}
#nav {width: 100%;height: 32px;}        /*定义列表框宽和高*/
#nav li {/*定义列表项样式*/
    float: left;                /*浮动列表项*/
    width: 9%;                  /*定义百分比宽度*/
    padding: 0 5%;              /*定义百分比补白*/
    margin: 0 2px;              /*定义列表项间隔*/
    background: #def;           /*定义列表项背景色*/
    font-size: 16px;
    line-height: 32px;          /*垂直居中*/
    text-align: center;         /*平行居中*/
}
</style>
<ul id="nav">
    <li>美 丽 说</li>
    <li>聚美优品</li>
    <li>唯 品 会</li>
    <li>蘑 菇 街</li>
```



```
        <li>1 号 店</li>
    </ul>
```



Note

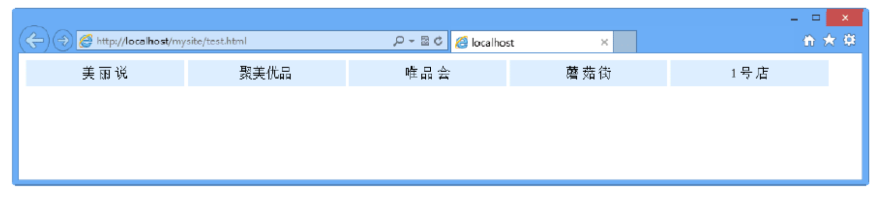


图 9.20　IE 下预览效果

在布局中，混用边界和补白来间隔不同模块区域或者分隔相邻元素。但下面这个问题应引起重视：当发生边界重叠或宽度溢出时，建议选用补白作为调整元素间距的首选属性。为了便于理解，请看下面示例。

```
<style type="text/css">
#box1 {margin-bottom: 6px;}
#box2 {padding-top: 3px;}
</style>

<div id="box1">上边元素</div>
<div id="box2">下边元素</div>
```

这是一个简单的示例，假设上下两个模块的间距为 6px，现在要调整 box1 与 box2 的间距为 9px，此时有 4 种方法可供选择：

- ☑ 增加 box1 的 margin-bottom 的属性为 9px，但可能会影响同行其他模块的布局。
- ☑ 增加 box1 的 padding-bottom 的属性为 3px，但会使上边元素过于上移而呈现突兀。
- ☑ 增加 box2 的 margin-top 的属性为 3px，但由于上下元素边界的重叠而无法实现。如果要增加 box2 的 margin-top 的属性为 9px，会存在更大的布局危险。
- ☑ 增加 box2 的 padding-top 的属性为 3px，这样就可以实现 box1 与 box2 之间的间距为 9px，但又不会影响同行其他元素的布局，整个页面也不会出现过大起伏错落。

补白与边界、边框一样都是可选的，并不是每个元素都必须全部设置，如果不设置这些属性，CSS 默认其值为 0。但是很多元素已经被浏览器预定了特定样式，如 body、p、h1～6、ul 等，这些预定义样式主要包括补白和边界的属性设置。当然，也可以重置 margin 和 padding 为 0，清除其中的预定义样式。为了快速开发，可以在页面设计之初，用通配选择符清除所有元素的补白和边界样式。

```
/*[清除所有元素的预定义样式]*/
* {
    margin:0;                    /*清除边界值*/
    padding:0;                   /*清除补白值*/
}
```

9.5　在线练习

本节专题练习 CSS3 的盒模型相关知识，以及应用技巧，感兴趣的读者可以扫码练习。



在线练习



第10章 使用 CSS 布局网页

网页版式一般通过栏目的行、列组合来设计，如单行版式、两行版式、三行版式、多行版式、单列版式、两列版式、三列版式等，根据网页效果确定，而不是根据 HTML 结构确定。也可以根据栏目显示性质进行设计，如流动布局、浮动布局、定位布局、混合布局等，或者根据网页宽度进行设计，如固定宽度、弹性宽度等。本章将具体讲解 CSS3 布局的基本方法。

【学习重点】

- 了解 CSS 布局基础。
- 设计流动布局
- 设计浮动布局。
- 设计定位布局。
- 灵活使用不同方法设计综合页面布局效果。



10.1 显示类型



Note

网页对象的显示类型可以使用 display 属性来显式定义。任何元素都可以通过 display 属性改变默认显示类型，因此也会改变该元素所对应的网页布局结构。

在 CSS2.1 中，display 属性共有 18 个选项值，详细说明如下。

☑ block：块状显示，在元素后面添加换行符，也就是说其他元素不能在其后面并列显示。
☑ none：隐藏显示，这与“visibility:hidden;”声明不同，“display:none;”声明不会为被隐藏的元素保留位置。
☑ inline：行内显示，在元素后面删除换行符，多个元素可以在一行内并列显示。
☑ inline-block：行内显示，但是元素的内容以块状显示，行内其他元素还会显示在同一行内。
☑ compact：紧凑的块状显示或基于内容之上的行内显示。
☑ marker：在容器对象之前或之后显示，该属性值必须与:after 和:before 伪元素一起使用。
☑ inline-table：具有行内特征的表格显示。
☑ list-item：具有块状特征的列表项目显示，并可以添加可选项目标志。
☑ run-in：块状显示或基于内容之上的行内显示。
☑ table：具有块状特征的表格显示。
☑ table-caption：表格标题显示。
☑ table-cell：表格单元格显示。
☑ table-column：表格列显示。
☑ table-column-group：表格列组显示。
☑ table-header-group：表格标题组显示。
☑ table-footer-group：表格页脚组显示。
☑ table-row：表格行显示。
☑ table-row-group：表格行组显示。

详细说明可以参考 CSS 参考手册。CSS3 新增了 box 显示类型，关于该技术话题请参阅第 9 章详细讲解。如果单从布局角度来分析，这些显示类型都可以划归为 block 和 inline 两种基本显示形态，其他类型都是这两种类型的特殊显示。其中真正能够应用并获得所有浏览器支持的取值只有 4 个：block、none、inline、list-item。

none 属性值表示隐藏并取消盒模型，这样元素所包含的内容就不会被浏览器解析和显示，同样这个盒子所包含的任何元素都会被浏览器忽略，不管它们是否被声明为其他属性。

list-item 属性值表示列表项目，其实质上也是块状显示，不过是一种特殊的块状类型，它增加了缩进和项目符号。

另外，还有一些比较有用的显示类型，如 table、table-cell、inline-block、inline-table 等，它们在特殊布局中具有重要的实用价值。

10.2 CSS 布局类型

CSS2.1 明确了两个模型，一个是众所周知的盒模型（Box model），CSS1 没有盒模型的概念，盒



模型的前身在 CSS1 里叫作面向盒的格式化模型。元素抽象为盒，以盒为对象设计思路清晰很多。CSS3 的盒模型丰富了更多属性。盒的产生，以及盒的定位就是 CSS2.1 定义的第二个模型：可视格式化模型（Visual Formatting Model）。CSS3 相关的布局标准实际上也在这个大的框架之下。

CSS 布局有明确标准始于 CSS2。虽然 CSS1 里有 Float 元素的定义，但它的设计不是为页面布局，只是为了实现图文绕排，所以 CSS1 里没有布局的概念，早期用 table 布局便顺理成章。CSS2 是 1998 年变为推荐标准的，按理此时应该普及新的 CSS 布局标准，但 table 布局的使用习惯一直保持到 2004 年才被 Jeffrey Zeldman 的一本书点醒。2005 年中文版《网站重构》出版，国内掀起重构浪潮。早期网页开发者是受网页制作软件，如 Frontpage 和 Dreamweaver 等束缚的，完全没有标准的概念。

CSS2.1 的布局分为 3 种：常规流（Normal Flow）、浮动（Float）和绝对定位（Absolute Position），这 3 种不能混用。如果代码里看到“position:absolute;display:block;”，这种明显是概念混乱。很多人也错把 position:relative 和 position:absolute 归为一类。position:relative 是常规流中的一种，例外是它可以和 Float 一起使用。

IE6 时代被忽视的常规流布局：

☑ Inline-Block。

兼容方法：触发 IE 的 hasLayout 可以实现相似的效果，以至于可以兼容 IE6 和 IE7，该方法逐渐被广泛应用。

☑ CSS Table。

兼容方法：http://caniuse.com/#feat=css-table，仅不兼容 IE6 和 IE7，在 IE6 时代被埋没。

对 Float 的滥用就像当年对 table 的滥用。很多人设计布局时不假思索地用 Float，明显欠缺对布局技术有更多了解。尤其在低端浏览器日渐淡出，新的布局技术触手可及的当下，是时候学习实践这些新技术了。

CSS3 只是一种笼统的叫法。有 CSS Level 1、CSS Level 2，时至今日还没有 CSS Level 3。CSS2.1 之上的新标准大部分仍在 WD 状态，只有 CSS Color Level 3 和 Selectors Level 3 进入 REC 状态。

提示：W3C 各状态说明请参阅第 1 章说明。

新的布局标准可以更简单、更灵活地实现布局。目前，CSS3 有如下多种布局方案。

1．Multi-Column Layout

当前状态：CR。

历史：1999 年 6 月 23 日发布 PD，2009 年 12 月 17 日进入 CR 状态。

兼容性：http://caniuse.com/#feat=multicolumn。

Demo：http://dabblet.com/gist/5507829。

问题：

☑ 目前只能平均分栏，还不支持分别指定栏宽（未来会有）。

☑ 浏览器支持的新旧标准不一。

例如，在上面 Demo 示例中，div.intro 的内容想保持在一栏中，Chrome 支持“column-break-inside: avoid;”，这是 2005 年 12 月 15 日更新的 WD 中的标准。由此推测 Chrome 遵循的是 2005 年的 WD 标准。目前 CR 标准已改成 break-inside。Firefox20 实现的仍是 2001 年的 WD 标准。

☑ 多栏布局更适合用于内容流布局，不适合页面布局。

2．Flexible Box Layout

当前状态：CR。

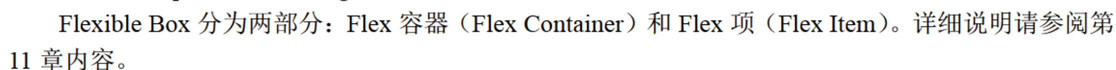
历史：2009 年 7 月 23 日发布 PD，2011 年 3 月 22 日进入 WD 状态，2012 年 9 月 18 日进入 CR



状态。

兼容性：http://caniuse.com/#feat=flexbox。

Demo：http://dabblet.com/gist/5508104。



Flexible Box 分为两部分：Flex 容器（Flex Container）和 Flex 项（Flex Item）。详细说明请参阅第 11 章内容。



Note

3．Template Layout

(Grid) Template Layout 曾经称为 Advanced Layout。

当前状态：WD。

历史：2005 年 12 月 15 日发布 PD，2007 年 8 月 9 日进入 WD 状态。最终合并进 Grid Layout。

4．Grid Position

历史：2007 年 9 月 5 日发布 PD，随后被废弃。

5．Grid Layout

当前状态：WD。

历史：2011 年 4 月 7 日发布 PD，2012 年 3 月 22 日进入 WD 状态，最新一版是 2013 年 4 月 2 日发布。

兼容性：http://caniuse.com/#feat=css-grid。

以这种兼容性还不值得介入学习。1996 年有个基于帧布局的草案，当时没往这个方向发展，最终绝对定位进入 CSS2。Grid Layout 正是建在它的基础之上。

6．CSS3 Floating Boxes

CSS3 Floating Boxes 被称为 CSS3 的浮动盒。

当前状态：WD。

历史：2002 年 10 月 24 日出现在 CSS basic box model 的草案中。

CSS3 的浮动盒技术过于超前，目前还没有浏览器支持。例如：

☑ "float: right contour;"，contour 关键字，文字可以沿图片不规则的轮廓绕排。

☑ "min-height: contain-floats;"，新的清浮动方式。

☑ "float-displace: block;"，新的 float-displace 属性。

7．Regions 和 Shapes

当前状态：WD。

历史：2011 年 6 月 9 日发布 PD，2011 年 11 月 29 日进入 WD 状态，最新一版是 2012 年 8 月 23 日发布。

兼容性：http://caniuse.com/#feat=css-regions。

Demo：http://dabblet.com/gist/5509294（Chrome 19+、IE10+）

它适用于内容流布局，兼容性有限，不建议提前学习。

10.3　流动布局

CSS 包含 3 种基本的布局模型，用英文概括为 Flow、Layer 和 Float，它们分别表示流动布局、定位布局（层）和浮动布局。流动模型（Flow Model）是 HTML 默认布局模型，默认状态下 HTML



元素都是根据流动模型分布，并随文档流自上而下按顺序动态分布。流动布局只能根据元素排列的先后顺序决定显示位置。如果要改变元素的显示位置，只能通过改变 HTML 文档结构实现。

流动布局模型的优势：元素之间不会存在错位、覆盖等问题，布局简单，符合浏览习惯。当然这种布局模型的弱点也是显而易见的，用户不能设计更灵活的版式效果。



10.3.1　流动元素

流动是默认的网页布局模式。任何没有定义“position:absolute;”或“position:fixed;”属性，以及没有定义“float:left;”或“float:right;”的元素默认都呈现为流动布局状态。

下面介绍流动布局模型比较典型的两个特征。

第一，块状元素都会在包含元素内自上而下按顺序垂直堆叠分布。在默认状态下，块状元素的宽度都为 100%，占据一行显示，不管这个元素是否包含内容，宽度是否为 100%。

【示例】下面示例设计在页面中添加多个对象，浏览器都会自上向下逐个解析并显示所有网页对象，如图 10.1 所示。

```
<div id="contain">
    <h2>标题元素</h2>
    <p>段落元素</p>
    <ul>
        <li>列表项</li>
    </ul>
    <table>
        <tr>
            <td>表格行，单元格</td>
            <td>表格行，单元格</td>
        </tr>
    </table>
</div>
```

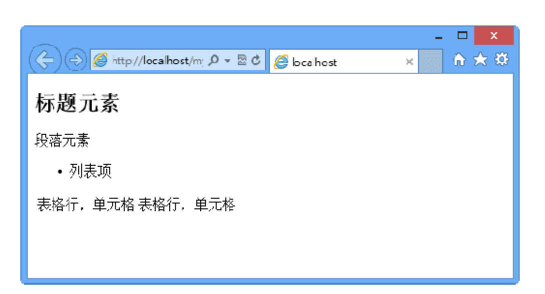


图 10.1　默认流动布局显示效果

第二，行内元素都会在包含元素内从左到右水平分布显示，有人把这种分布方式称为文本流，文本流源于文本的从左向右自然流动，本质上它与 HTTP 传输方式和浏览器的解析机制密切相关。超出一行后，会自动换行显示，然后继续从左到右按顺序流动，以此类推。



10.3.2　相对定位元素

当元素定义为相对定位，即设置“position:relative;”属性时，它也会遵循流动模型布局，跟随 HTML 文档流自上而下流动。





Note

【示例 1】下面示例借用 10.3.1 节的文档结构，然后单独定义 p 元素以相对定位显示，它会严格遵循流动模型，自上而下按顺序流动显示，如图 10.2 所示。这是一个非常重要的特征，在实践中可以利用这一特征解决定位布局难题。

```
<style type="text/css">
#contain {border: double 4px #999;}              /*定义一个包含框*/
#contain h2 {background: #FFCCCC;}               /*定义标题背景色*/
#contain p {/*定义段落属性*/
    border-bottom: solid 1px #999;
    position: relative;                          /*设置段落元素为相对定位*/
}
#contain table {border: solid 1px #00CCFF;}      /*定义表格边框*/
</style>

<div id="contain">
    <h2>标题元素</h2>
    <p>段落元素</p>
    <ul>
        <li>列表项</li>
    </ul>
    <table>
        <tr>
            <td>表格行，单元格</td>
            <td>表格行，单元格</td>
        </tr>
    </table>
</div>
```

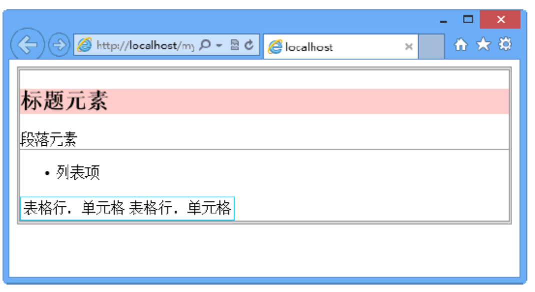


图 10.2　相对定位流动显示效果

【示例 2】在下面示例中定义 p 元素相对偏移，这时会发现它偏离原来的位置，但依然遵循流动模型规则，始终保持与原点相同的位置关系，随文档流整体移动。

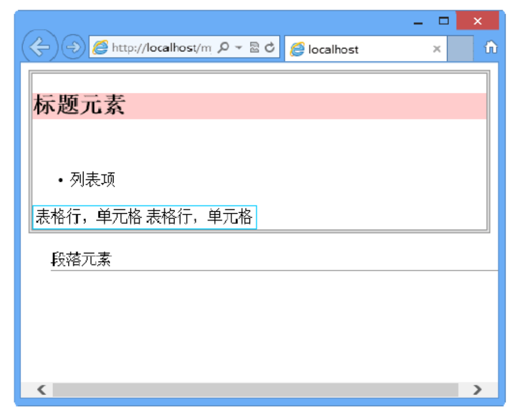
如果在标题元素前面增加多行空行，相对定位的 p 元素会随其他元素一起向下移动，同时也会看到 p 元素原位置与偏移后的位置始终保持不变，如图 10.3 所示。

```
<style type="text/css">
#contain {border: double 4px #999;}              /*定义一个包含框*/
#contain h2 {background: #FFCCCC;}               /*定义标题背景色*/
#contain   p {/*定义段落元素属性*/
    border-bottom:solid 1px #999;
    position:relative;                           /*设置段落元素为相对定位*/
```

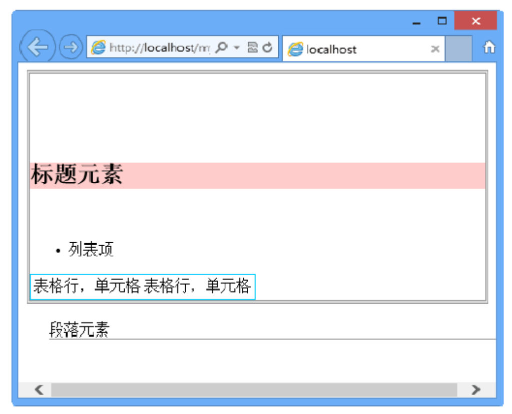



```
        left:20px;                              /*以原位置左上角为参考点向右偏移 20px*/
        top:120px;                              /*以原位置左上角为参考点向下偏移 120px*/
    }
    #contain table {border: solid 1px #00CCFF;}  /*定义表格边框*/
    </style>

    <div id="contain">
        <br><br><br><br>
        <h2>标题元素</h2>
        <p>段落元素</p>
        <ul>
            <li>列表项</li>
        </ul>
        <table>
            <tr>
                <td>表格行，单元格</td>
                <td>表格行，单元格</td>
            </tr>
        </table>
    </div>
```



相对偏移位置



随文档流下沉

图 10.3　相对定位流动显示效果

> **提示：** 如果一个绝对定位元素没有明确定义 left 或 right，则它也会随文档流在水平方向上移动；如果一个绝对定位元素没有明确定义 top 或 bottom，则它也会随文档流在垂直方向上移动。

10.4　浮 动 布 局

浮动模型（Float Model）汲取了流动布局和定位布局的优点，希望在流动与固定之间取得一个平衡，实现网页布局的自适应能力。层布局模型的问题是浏览器永远都是一个活动的窗口，无法预知不同浏览者所要使用的窗口大小，同时大部分网页大小本身也是无法预测和控制的。



浮动布局不同于流动模式，元素能够脱离左右相邻元素，在包含框内向左或右侧浮动显示，但是浮动元素不能脱离文档流，依然受文档流的影响。



Note



视频讲解

10.4.1 定义浮动显示

在默认情况下任何元素不具有浮动特性，可以使用 CSS 的 float 属性定义元素向左或向右浮动，基本用法如下：

```
float: none | left | right
```

其中 none 表示消除浮动，left 表示元素向左浮动，right 表示元素向右浮动，默认值为 none。

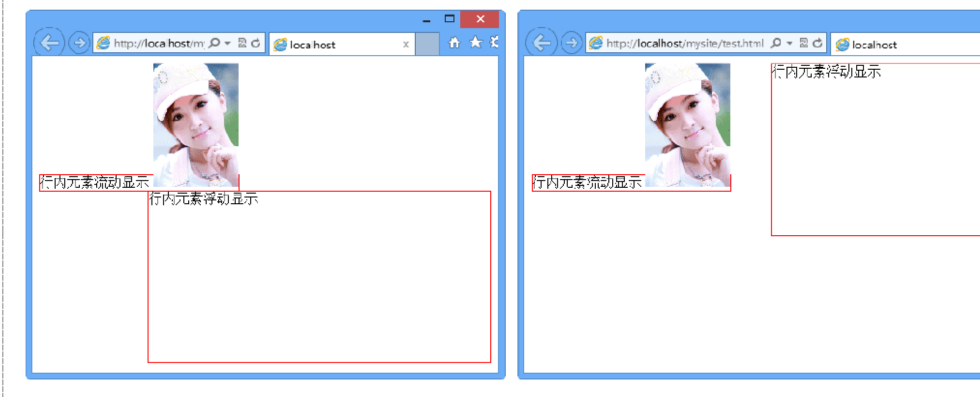
下面介绍浮动布局模型的几个特征。

第一，浮动元素以块状显示，可以定义 width 和 height 属性。

【示例 1】下面示例为两个 span 元素定义高和宽属性，然后让其中一个 span 元素浮动显示，来比较它们的显示效果，如图 10.4 所示。

```
<style type="text/css">
span {/*定义行内元素 span 的显示属性*/
    width:400px;                    /*定义宽为 400px*/
    height:200px;                   /*定义高为 200px*/
    border:solid red 1px;}
#inline img {width:100px;}          /*定义行内元素内的图片宽为 100px*/
#float {float:right;}               /*为第 2 个行内元素 span 定义浮动显示*/
</style>

<span id="inline">行内元素流动显示
    <img src="images/1.jpg" alt="流动的图片" />
</span>
<span id="float">行内元素浮动显示</span>
```



行内元素浮动显示 1　　行内元素浮动显示 2

图 10.4　浮动显示与流动显示比较

通过图 10.4 可以看到，当第 2 个 span 元素被定义为浮动之后，该元素自动以块状显示，因此为 span 元素定义的高和宽属性值有效。而第 1 个元素由于是行内元素且没有浮动显示，所以定义的宽和高无效，所看到的红色边框仅包裹在行内元素的外边。

浮动元素应该明确定义大小。如果浮动元素没有定义宽度和高度，它会自动收缩到仅能包住内容



为止。例如，如果浮动元素内部包含一张图片，则浮动元素将与图片一样宽，如果是包含的文本，则浮动元素将与最长文本行一样宽。而当块状元素没有定义宽度时，则会自动显示为 100%。

第二，浮动元素与流动元素可以混合使用，不会重叠，都遵循先上后下显示规则，都受到文档流影响。但浮动元素能够改变相邻元素的显示位置，可以向左或向右并列显示。



Note

与普通元素一样，浮动元素始终位于包含元素内，不会脱离包含框，这与定位元素不同。

【示例 2】下面示例以示例 1 为基础，然后添加一个包含框，可以看到第 2 个 span 元素靠近包含元素 div 的右边框浮动，而不再是 body 元素的右边框，如图 10.5 所示。

```
<div id="contain">
    <span id="inline">行内元素流动显示
        <img src="images/1.jpg" alt="流动的图片" />
    </span>
    <span id="float">行内元素浮动显示</span>
</div>
```

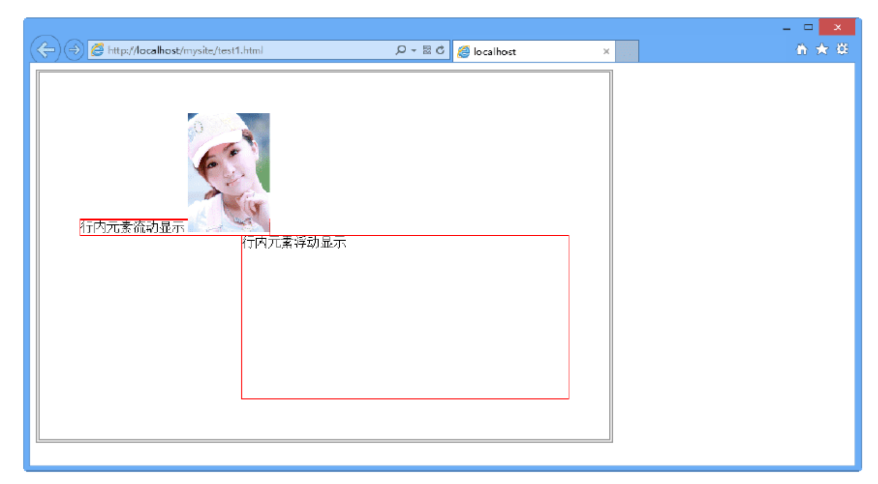


图 10.5　浮动元素始终位于包含元素

第三，浮动元素仅能改变水平显示方式，不能改变垂直显示方式，依然受文档流影响。流动元素总会以流动的形式环绕浮动元素左右显示。

浮动元素不会强迫前面的流动元素环绕其周围流动，而总是在上邻流动元素的下一行浮动显示。浮动元素不会覆盖其他元素，也不会挤占其他元素的位置。

第四，浮动元素可以并列显示，如果包含框宽度不够，则会错行显示。

【示例 3】下面示例模拟设计了 3 个并列显示的栏目，通过 float 定义左、中、右 3 栏并列显示，效果如图 10.6 所示。

```
<style type="text/css">
body {padding: 0; margin: 0; text-align: center;}
#main {/*定义网页包含框样式*/
    width: 400px;
    margin: auto;
    padding: 4px;
    line-height: 160px;
    color: #fff;
```





Note

```
        font-size: 20px;
        border: solid 2px red;}
#main div {float: left;height: 160px;}              /*定义 3 个并列栏目向左浮动显示*/
#left {width: 100px;background: red;}               /*定义左侧栏目样式*/
#middle {width: 200px;background: blue;}            /*定义中间栏目样式*/
#right {width: 100px; background: green;}           /*定义右侧栏目样式*/
.clear {clear: both;}
</style>

<div id="main">
        <div id="left">左侧栏目</div>
        <div id="middle">中间栏目</div>
        <div id="right">右侧栏目</div>
        <br class="clear" />
</div>
```

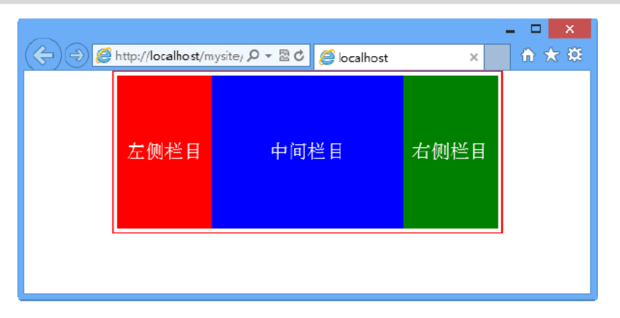


图 10.6　并列浮动显示

浮动布局可以设计多栏并列显示效果，但也容易错行，如果浏览器窗口发生变化或者浮动包含框不固定，则会出现错行浮动显示问题，破坏并列布局效果。



视频讲解

10.4.2　清除浮动

使用 CSS 的 clear 属性可以清除浮动，定义与浮动相邻的元素在必要的情况下换行显示，这样可以控制浮动元素挤在一行内显示。clear 属性取值包括 4 个。

☑　left：清除左边的浮动元素，如果左边存在浮动元素，则当前元素会换行显示。

☑　right：清除右边的浮动元素，如果右边存在浮动元素，则当前元素会换行显示。

☑　both：清除左右两边浮动元素，不管哪边存在浮动对象，则当前元素都会换行显示。

☑　none：默认值，允许两边都可以存在浮动元素，当前元素不会主动换行显示。

【示例】下面示例设计一个简单的 3 行 3 列页面结构，设置中间 3 栏平行浮动显示，如图 10.7 所示。

```
<style type="text/css">
div {
        border: solid 1px red;              /*增加边框，以方便观察*/
        height: 50px;                       /*固定高度，以方便比较*/
}
#left, #middle, #right {
        float: left;                        /*定义中间 3 栏向左浮动*/
        width: 33%;                         /*定义中间 3 栏等宽*/
```






```
    }
    </style>

    <div id="header">头部信息</div>
    <div id="left">左栏信息</div>
    <div id="middle">中栏信息</div>
    <div id="right">右栏信息</div>
    <div id="footer">脚部信息</div>
```

如果设置左栏高度大于中栏和右栏高度，则发现脚部信息栏上移并环绕在左栏右侧，如图 10.8 所示。

```
#left {height:100px;}                    /*定义左栏高出中栏和右栏*/
```

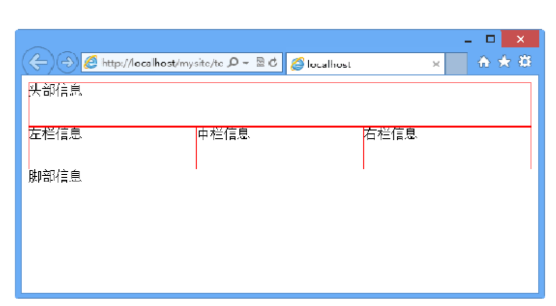


图 10.7　IE 6 中浮动布局效果

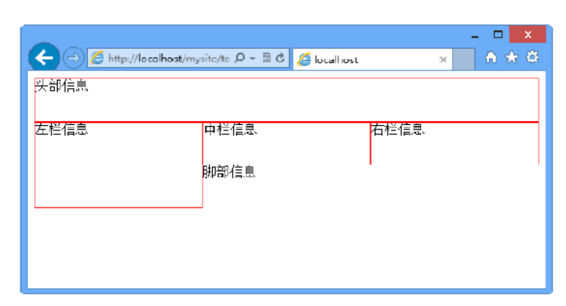
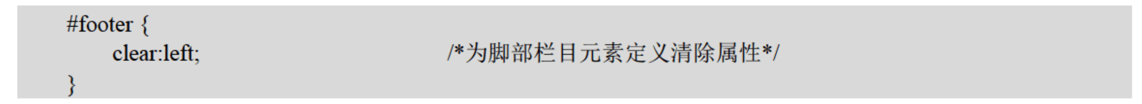


图 10.8　调整部分栏目高度后发生的错位现象

这时 clear 属性就可以派上用场了，为<div id="footer">元素定义一个清除样式：



```
#footer {
    clear:left;                          /*为脚部栏目元素定义清除属性*/
}
```

这时在浏览器中预览，则又恢复到预设的 3 行 3 列布局效果，如图 10.9 所示。

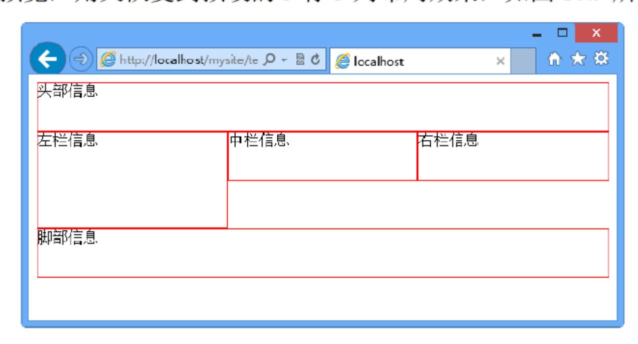


图 10.9　清除浮动元素错行显示

提示： clear 属性是专门针对 float 属性而设计的，因此仅能够对左右两侧浮动元素有效，对于非浮动元素是无效的。

清除不是清除浮动元素，而是清除自身，也即不允许当前元素与浮动元素并列显示。如果左右两侧存在浮动元素，则当前元素把自己清除到下一行显示，而不是把前面的浮动元素清除到下一行或者上一行显示。


视频讲解


Note

10.4.3 浮动嵌套

浮动元素可以相互嵌套，嵌套规律与流动元素嵌套相同。浮动的包含元素总会自动调整自身高度和宽度以实现对浮动子元素的包含。

【示例 1】新建文档，构建两个简单的嵌套块，然后强制它们浮动显示，并定义子元素的高度和宽度，使其显示为一定大小的区域，这时会发现父元素自动调整自身大小来包含子元素，如图 10.10 所示（test.html）。

```
<style type="text/css">
.wrap {border: solid 2px red; float: left; margin:4px;}
.sub {width: 200px; height: 200px; float: left; background: blue;}
</style>

<div class="wrap">
    <div class="sub"></div>
</div>
<span class="wrap">
    <span class="sub"></span>
</span>
```

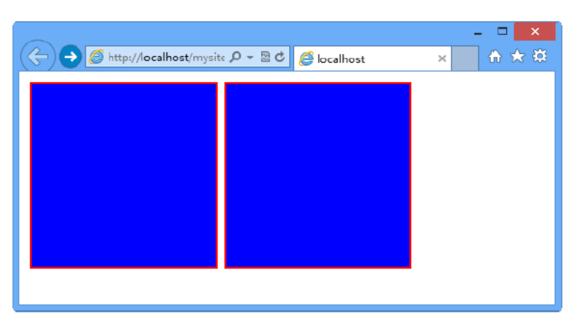


图 10.10　浮动嵌套

提示：如果包含元素定义了高度和宽度，则它就不会随子元素的大小而自动调整自身显示区域来适应子元素的显示，如图 10.11 所示（test1.html）。注意，在 IE6 及更低版本浏览器中包含框仍然能够自动调整自身大小来适应子元素的显示大小，不过在 IE7 版本中微软纠正了这个不符合标准的显示方法。

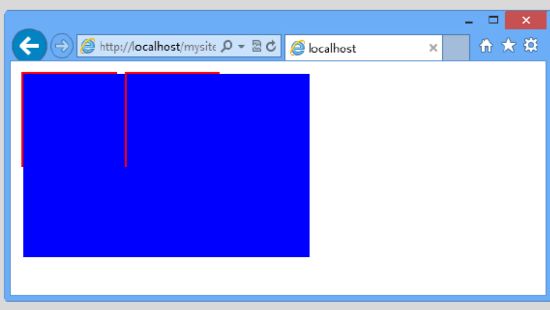


图 10.11　浮动嵌套存在问题



【示例 2】如果把浮动元素嵌入流动元素之内，则父元素不能够自适应子浮动元素的高度，如图 10.12 所示（test2.html）。

```
<style type="text/css">
#contain {background: #FF99FF;}/*包含元素*/
span {float: left; width: 200px; height: 100px;}/*定义共同属性*/
/*内嵌浮动对象样式*/
#span1 {border: solid blue 10px;}
#span2 {border: solid red 10px;}
</style>

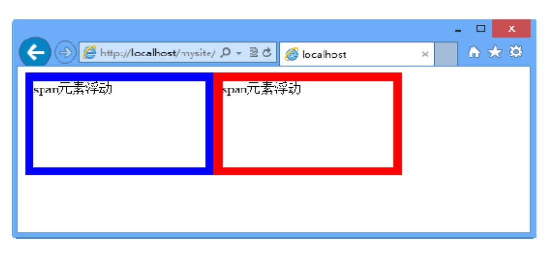
<div id="contain">
    <span id="span1">span 元素浮动</span>
    <span id="span2">span 元素浮动</span>
</div>
```

在图 10.12 中可以看到包含元素 div 并没有显示。原因就是包含元素没有适应子元素的高度，而是根据自身定义的属性以独立的形式显示。所以，在应用混合嵌套时，要预测到浮动与流动混合布局时会出现的各种现象，并积极做好兼容处理。

解决方法：可以在包含元素内的最后一行添加一个清除元素，强制撑开包含元素，使其包含浮动元素，显示结果如图 10.13 所示（test3.html）。

```
<style type="text/css">
#contain {background: #FF99FF;}/*包含元素*/
span {float: left; width: 200px; height: 100px;}/*定义共同属性*/
/*内嵌浮动对象样式*/
#span1 {border: solid blue 10px;}
#span2 {border: solid red 10px;}
.clear {/*定义清除类*/
    clear:both;}
</style>

<div id="contain">
    <span id="span1">span 元素浮动</span>
    <span id="span2">span 元素浮动</span>
    <div class="clear"></div><!--增加一个清除元素-->
</div>
```



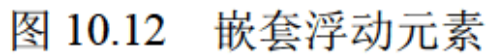


图 10.12　嵌套浮动元素

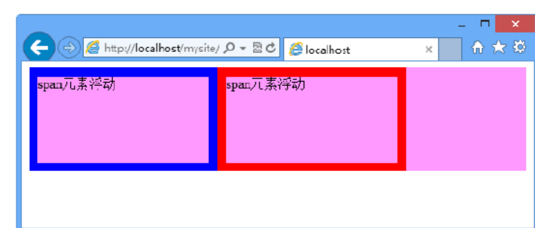
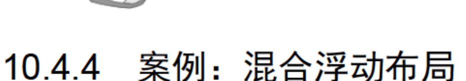


图 10.13　正确显示效果





视频讲解



Note

10.4.4 案例：混合浮动布局

浮动布局模型相比流动布局模型要复杂很多，当混合浮动和流动布局时就容易遇到很多问题，下面结合示例介绍常见问题和解决办法。

1. 调整左右栏间距

【示例 1】 制作一个左右两栏的页面，左栏是浮动布局，右栏是流动布局，显示结果如图 10.14 所示（test.html）。

```
<style type="text/css">
#contain {/*页面布局包含元素*/
    width:774px;                    /*定义页面宽*/
    border:double 4px #aaa;         /*定义页面边框*/
    padding:12px;                   /*为页面包含元素增加补白*/
    overflow:visible;               /*定义包含元素自动伸缩显示所有包含内容*/
}
#contain img {/*定义左侧图片浮动显示*/
    width:200px;
    height:100px;
    float:left;
    clear:left;                     /*定义图片单列显示*/
    margin:0 12px 6px 0;            /*定义图片的边界*/
    padding:6px;
    border:solid 1px #999;}
#contain h2 {/*定义右侧标题居中*/
    text-align:center;}
#contain p {/*定义段落属性*/
    margin:0;
    padding:0;
    line-height:1.8em;
    font-size:13px;
    text-indent:2em;}
.clear {/*定义清除类，处理非 IE 浏览器不能自适应包容问题*/
    clear:both;}
</style>

<div id="contain">
    <img src="bg10.jpg" />
    <img src="bg7.jpg" />
    <img src="bg2.jpg" />
    <img src="bg3.jpg" />
    <h2>《荷塘月色》（节选）</h2>
    <p>曲曲折折的荷塘上面，弥望的是田田的叶子。叶子出水很高，像亭亭的舞女的裙。…</p>
    <div class="clear"></div>
</div>
```

上面示例在图文混排基础上利用浮动模型设计一个更漂亮的图片通栏布局。使用 float 属性定义所有图片向左浮动，定义 clear 属性清除相邻图片并列浮动，将一组图片垂直排列。通过定义左栏浮



动、右栏流动，以保证页面中的文本围绕图片在右侧显示。



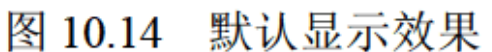


图 10.14　默认显示效果

如果加大右侧文本与左侧图片之间的间距，一般用户会定义 p 元素的 margin-left 或 padding-left 属性值，但会发现为 p 元素定义左边界或左补白之后，左右栏间距没有变。

解决方法：不要定义流动元素的边界或补白，而是定义浮动元素的边界或补白，实现调控间距的目的。因为浮动元素的边界和补白不会被流动元素覆盖。例如，定义浮动图像的右侧边界为 50px，则效果如图 10.15 所示（test1.html）。

```
margin:0 50px 6px 0;
```



图 10.15　调控间距显示效果

2．调整上下栏间距

上下栏之间的浮动与流动混合布局也比较复杂，如上下栏间距不易调整、布局偶尔错乱等。





Note

【示例 2】以示例 1 为基础上，给图文页面增加一个导航条，显示结果如图 10.16 所示（test2.html）。

```
<style type="text/css">
body {/*定义窗口属性*/
      margin:0;                    /*清除 IE 默认边界属性值*/
      padding:0;                   /*清除非 IE 默认补白属性值*/
}
#nav {/*定义导航列表框属性*/
      margin:0;                    /*清除 IE 默认缩进属性值*/
      padding:0;                   /*清除非 IE 默认缩进属性值*/
      list-style-type:none;        /*清除浏览器默认列表样式*/
}
#nav li {/*定义菜单列表项显示效果*/
      float:left;                  /*向左浮动*/
      width:100px; height:32px;
      line-height:32px;            /*垂直居中*/
      text-align:center;           /*水平居中*/
      background:#7B9F23;          /*背景色*/
      margin:1px;                  /*菜单间距*/
      font-size:14px;}
#nav a {text-decoration:none;}     /*定义导航链接属性*/
#contain {/*图文包含元素*/
      width:774px;                 /*定义图文框宽*/
      border:double 4px #aaa;      /*定义图文框边框*/
      padding:12px;                /*为图文框增加补白*/
      overflow:visible;            /*定义图文框自动伸缩显示所有包含内容*/
}
#contain img {/*定义左侧图片浮动显示*/
      width:200px; height:100px;
      float:left; clear:left;      /*定义图片单列显示*/
      margin:0 12px 6px 0;         /*定义图片的边界*/
      padding:6px; border:solid 1px #999;}
#contain h2 {text-align:center;}   /*定义右侧标题居中*/
#contain p {/*定义段落属性*/
      margin:0; padding:0;
      line-height:1.8em; font-size:13px;
      text-indent:2em;}
.clear {clear:both;}               /*定义清除类，处理非 IE 浏览器不能自适应包容问题*/
</style>

<ul id="nav"><!--导航菜单模块-->
      <li><a href="">首页</a></li>
      <li><a href="">导航菜单 1</a></li>
      <li><a href="">导航菜单 2</a></li>
      <li><a href="">导航菜单 3</a></li>
      <li><a href="">导航菜单 4</a></li>
      <li><a href="">导航菜单 5</a></li>
      <li><a href="">导航菜单 6</a></li>
</ul>
<div id="contain"><!—图文框模块-->
```



```
    ……
</div>
```



图 10.16　给页面增加导航条

通过图 10.16 可以发现导航条错位显示在下面栏目内部。解决方法：在列表项最后添加一个清除元素：

```
<div class="clear"></div>
```

上述代码强迫 ul 元素自适应高度，以实现包含其内部的浮动列表项。这样就不会出现浮动元素与流动包含元素相互脱节现象，使浮动元素显示在上面栏目包含框中，效果如图 10.17 所示（test3.html）。



图 10.17　正确显示效果



10.5 定 位 布 局



定位模型也称层模型（Layer Model），最早源于 Netscape Navigator 4.0 推出并支持的 Layer（层）。Layer 可以将页面的内容引入层的概念，模拟 Photoshop 中的层处理技术，摆脱 HTML 元素自然流动带来的弊端，增强网页处理能力。

定位布局的设计思路比较简单，它允许用户精确定义网页元素的显示位置，可以是绝对位置，也可以是相对位置，这里的相对可以是相对元素原来显示的位置，也可以是相对最近的定位包含框元素，或者是相对浏览器窗口。



视频讲解

10.5.1 定义定位显示

在 CSS 中可以通过 position 属性定义元素定位显示，其语法如下：

```
position: static | relative | absolute | fixed
```

取值说明如下。

☑ static：表示不定位，元素遵循 HTML 默认的流动模型，如果未显式声明元素的定位类型，则默认为该值。

☑ relative：表示相对定位，它通过 left、right、top、bottom 属性确定元素在正常文档流中偏移位置。相对定位完成的过程是首先按 static 方式生成一个元素，然后移动这个元素，移动方向和幅度由 left、right、top、bottom 属性确定，元素的形状和偏移前的位置保留不动。

☑ absolute：表示绝对定位，将元素从文档流中拖出来，然后使用 left、right、top、bottom 属性相对于其最接近的一个具有定位属性的父定位包含框进行绝对定位。如果不存在这样的定位包含框，则相对于浏览器窗口，而其层叠顺序则通过 z-index 属性来定义。

☑ fixed：表示固定定位，与 absolute 定位类型类似，但它的定位包含框是视图本身，由于视图本身是固定的，它不会随浏览器窗口的滚动条滚动而变化，除非在屏幕中移动浏览器窗口的屏幕位置或改变浏览器窗口的显示大小，因此固定定位的元素会始终位于浏览器窗口内视图的某个位置，不会受文档流动影响，这与“background-attachment:fixed;”属性功能相同。

与浮动元素一样，绝对定位元素以块状显示，它会为所有子元素建立一个定位包含框，所有被包含元素都以定位包含框作为参照物进行定位，或在其内部浮动和流动。

【示例 1】在下面示例中，定义了 3 个不同模型的包含元素，然后观察不同模型的包含元素与它们的子元素的位置关系，如图 10.18 所示。

```
<style type="text/css">
#contain1, #contain2, #contain3 {/*定义 3 个一级 div 元素对象的共同属性*/
    width: 380px;height: 120px; border: solid 1px #666;}
#contain2 {/*定义第 2 个一级 div 元素对象为绝对定位，并设置其距离窗口左边和上边的距离*/
    position: absolute; left: 120px; top: 60px; background: #F08080;}
#contain3 {/*定义第 3 个一级 div 元素对象为浮动布局*/
    float: left; background: #D2B48C;}
#contain2 div {/*定义绝对定位对象内所有子元素对象的共同属性*/
    color: #993399; border: solid 1px #FF0000;}
#sub_div1 {/*定义绝对定位包含框内第 1 个对象为绝对定位*/
```



```
        width: 80px; height: 80px; position: absolute;
        right: 10px; /*定义该绝对元素右边距离父级定位包含框的右边距离*/
        bottom: 10px; /*定义该绝对元素底边距离父级定位包含框的底边距离*/
        background: #FEF68F;}
    #sub_div2 {/*定义绝对定位包含框内第 2 个元素为浮动布局*/
        width: 80px; height: 80px; float: left; background: #DDA0DD;}
    #sub_div3 {/*定义绝对定位包含框内第 3 个元素的背景色、宽和高*/
        width: 100px; height: 90px; background: #CCFF66;}
    </style>

    <div id="contain1">元素一流动</div>
    <div id="contain2">元素二－绝对定位
        <div id="sub_div1">子元素 1－绝对定位</div>
        <div id="sub_div2">子元素 2－浮动</div>
        <div id="sub_div3">子元素 3－流动</div>
    </div>
    <div id="contain3">元素三－浮动</div>
```



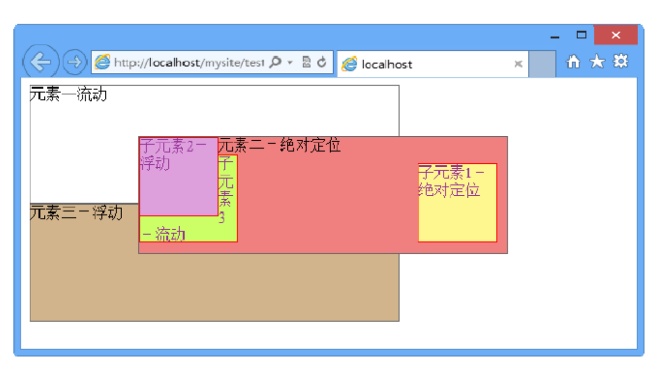


图 10.18　定位显示

从图 10.18 可以看到“元素二”被定义为绝对定位，它以浏览器窗口为定位包含框，显示位置根据元素左边到窗口左边的距离和元素上边到窗口上边的距离来确定。而“子元素 1”却以具有定位属性的“元素二”为定位包含框，显示位置根据“子元素 1”右边到“元素二”右边的距离和元素底边到“元素二”底边的距离来确定。在“元素二”中包含了 3 个子元素，它们以不同的性质显示，但它们都以“元素二”作为参照平台，包括其中的浮动元素和绝对定位元素。

如果把行内元素作为定位包含框，情况就很复杂，因为行内元素有可能在几行内显示，产生好几个线性盒，这时定位包含框就被定义为这几行区域，而其内部被包含的绝对定位子元素将根据行内元素的第 1 行第 1 个字符左上角来确定 left 和 top 属性的偏移值，根据第 1 行最后 1 个字符的右下角来确定 right 和 bottom 属性的偏移值。

【示例 2】在下面示例中，在文本段中设置两个相互嵌套的 span 元素，然后把外层的 span 元素定义为定位包含框，而把内层的 span 元素定义为绝对定位，并进行偏移定位，显示效果如图 10.19 所示。

```
    <style type="text/css">
    p {/*定义文本段属性*/
        width: 400px; height: 200px;
        border: dashed 1px green;}
    #relative {/*定义定位包含框，并用蓝色线框标示*/
```



Note

```
        position: relative;
        border: solid 1px blue;}
    #absolute {/*定义绝对定位子元素，并向右下角偏移 200px*/
        position: absolute;
        left: 200px; top: 200px;
        border: solid 2px red;}
    </style>

    <p> 在用 CSS 控制排版过程中，<span id="relative">定位一直被人认为是一个难点，这主要是表现为很多网友在没有深入理解清楚定位的原理时，排出来的杂乱网页常让他们不知所措，而另一边一些高手则常常借助定位的强大功能做出些很酷的效果来，<span   id="absolute">比如 CSS 相册等等</span>，因此自己杂乱的网页与高手完美的设计形成鲜明对比，</span>这在一定程度上打击了初学定位的网友，也在他们心目中形成这样的一种思想：当我熟练地玩转 CSS 定位时，我就已是高手了。</p>
```

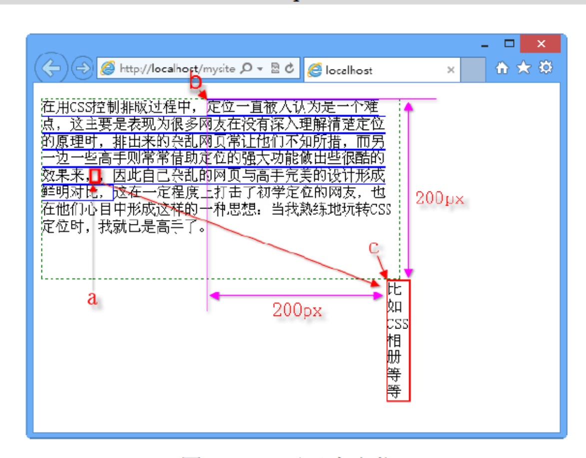


图 10.19　子元素定位 1

在图 10.19 中，a 所指示的位置为子元素定位前的位置，b 所指示的顶角为子元素偏移的参考点，c 所指示的顶角为偏移后的定位点。可以看到，在定位包含框多行显示时，内部定位元素将根据 b 点（第 1 行第 1 个字的左上角）进行偏移定位，而不是以文本段左上角作为参考点进行定位。

如果定义 right 和 bottom 属性进行偏移，其中 CSS 代码如下，内部定位元素将根据 b 点（第 1 行最后 1 个字的右下角）进行偏移定位，而不是以文本段右下角作为参考点进行定位。

```
    #absolute {/*定义绝对定位子元素，并向左上角偏移 100px*/
        position:absolute;
        right:100px;
        bottom:100px;
        border:solid 2px red;}
```



视 频 讲 解

10.5.2　相对定位

与绝对定位不同的是，相对定位元素的偏移量是根据它在正常文档流里的原始位置计算的，而绝对定位元素的偏移量是根据定位包含框的位置计算的。一个绝对定位元素的位置取决于它的偏移量：top、right、bottom 和 left 属性值，相对定位元素的偏移量与绝对定位一样。

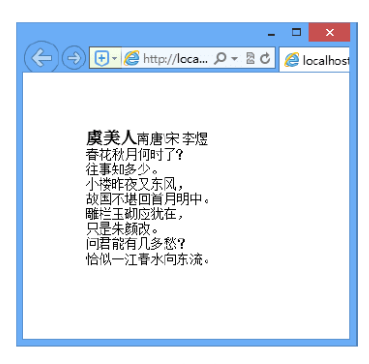
【示例】在下面示例中，定义 strong 元素对象为相对定位，然后通过相对定位调整标题在文档顶部的显示，显示效果如图 10.20 所示。



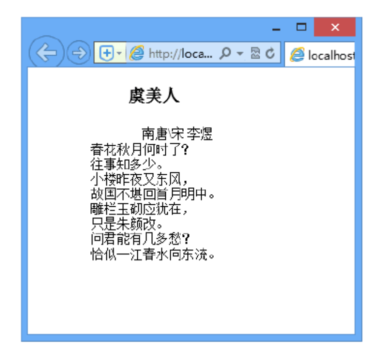



```
<style type="text/css">
p {margin: 60px; font-size: 14px;}
p span {position: relative;}
p strong {/*[相对定位]*/
    position: relative;
    left: 40px; top: -40px;
    font-size: 18px;}
</style>

<p> <span><strong>虞美人</strong>南唐\宋 李煜</span> <br>春花秋月何时了？<br>往事知多少。<br>小楼昨夜又东风，<br>故国不堪回首月明中。<br>雕栏玉砌应犹在，<br>只是朱颜改。<br>问君能有几多愁？<br>恰似一江春水向东流。 </p>
```



定位前



定位后

图 10.20　相对定位显示效果

从图 10.20 可以看到，相对定位后，元素对象的原空间保留不变。相对定位偏离的边距遵循绝对定位中偏离规则，不过相对定位的定位包含框是元素对象的原位置。

提示： 相对定位元素遵循的是流动布局模型，存在于正常的文档流中，但是它的位置可以根据原位置进行偏移。由于相对定位元素占有自己的空间，即原始位置保留不变，因此它不会挤占其他元素的位置，但可以覆盖在其他元素之上进行显示。

与相对定位元素不同，绝对定位元素完全被拖离正常文档流中原来的空间，且原来空间将不再被保留，而被相邻元素挤占。把绝对定位元素设置在可视区域之外会导致浏览器窗口的滚动条出现。而设置相对定位元素在可视区域之外，滚动条不会出现。

10.5.3　定位框



CSS 定位包含框是标准布局中一个重要概念，它是绝对定位的基础。注意区分定位包含框与父元素、包含框或包含元素等概念。

定位包含框就是为绝对定位元素提供坐标偏移和显示范围的参照物，即确定绝对定位的偏移起点和百分比长度的参考。在默认状态下，body 元素就是一个根定位包含框，所有绝对定位的元素就是根据窗口来确定自己所处的位置和百分比大小显示的。但是如果定义了包含元素为定位包含框，对于被包含的绝对定位元素来说，就会根据最接近的具有定位功能的上级包含元素来决定自己的显示位置。

【示例】 为了能直观理解定位包含框，下面示例先构建一个 HTML 代码模块。





```
<div id="a">
    <div id="c"></div>
</div>
<div id="b">
    <div id="d"></div>
</div>
```

在上面代码中，构建了两个定位包含框，它们分别包含了一个元素。下面用 CSS 定义这两个包含元素的大小为 200px *200px，并浮动在窗口的中间区域。

```
#a,#b {/*定义包含元素的共同属性*/
    width:200px;
    height:200px;
    float:left;
    margin-top:50px;                /*拉开与窗口顶部的距离*/
    border:solid 1px red;           /*定义红色边框线，便于识别*/
}
```

同时，单独定义 b 包含元素为相对定位，确定它是一个定位包含框。

```
#b {/*定义包含元素 b 为相对定位，确定它为定位包含框*/
    position:relative;
    margin-left:50px;               /*拉开与 b 包含元素的距离*/
}
```

然后，定义两个被包含元素为绝对定位，大小为 50%*50%，并都偏移 50%。

```
#c,#d {/*定义被包含元素为绝对定位，并进行偏移*/
    width:50%;
    height:50%;
    position:absolute;
    left:50%;                       /*与定位包含框左侧边框距离为 50%*/
    top:50%;                        /*与定位包含框顶部边框距离为 50%*/
}
```

最后，分别为两个被包含元素定义不同背景颜色，以便于区别，则显示如图 10.21 所示。

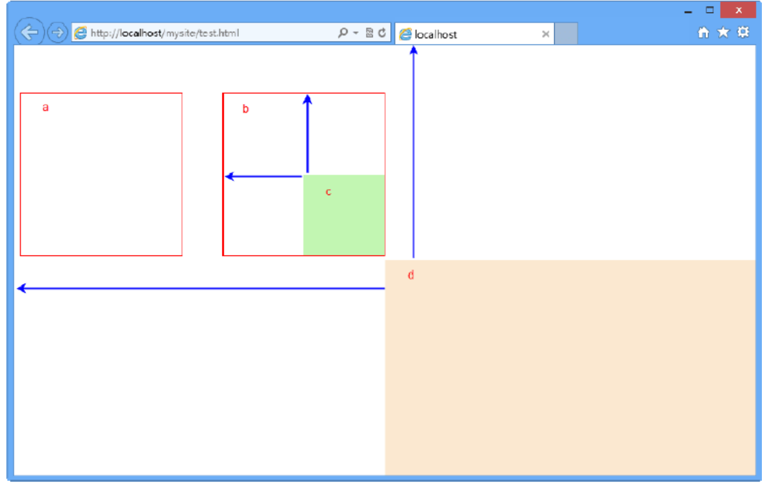
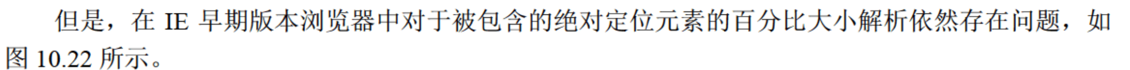


图 10.21　定义定位包含框







在图 10.21 所示的演示效果中，被 a 包含元素包含的 c 子元素，以窗口 body 元素的左上点为坐标原点进行绝对定位偏移，百分比大小取值也根据窗口的大小来确定，即为窗口宽度和高度的一半。

而 b 包含元素被定义为相对定位，它就成了一个定位包含框，因此，被它包含的 d 元素就会以 b 元素的左上角为坐标原点进行绝对定位偏移，其百分比大小取值也会根据 b 元素的大小来确定，而不是以窗口为参照物。

但是，在 IE 早期版本浏览器中对于被包含的绝对定位元素的百分比大小解析依然存在问题，如图 10.22 所示。

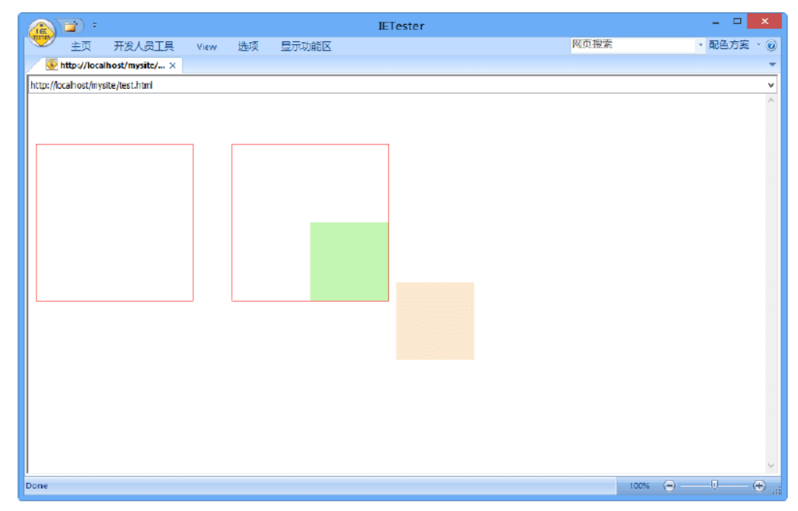


图 10.22　IE 中的定位包含框显示效果

图 10.22 是上面示例在 IE6 浏览器中的预览效果，可以看到，对于坐标偏移解析方面，IE 与其他现代标准浏览器的解析效果是一致的，即 a 包含元素内的 c 元素根据最近定位包含框（窗口左上角）进行偏移，百分比偏移大小（50%）也是根据定位包含框大小（窗口大小）来确定的。但是，在计算被包含元素自身大小时，IE6 与标准存在很大的差异，IE6 浏览器认为被包含元素 c 的百分比高和宽应该根据 HTML 代码中包含它的元素的大小来确定，而不是它的最近定位包含框，因此 c 元素显示大小（100px*100px）就为 a 元素显示大小（200px*200px）的四分之一。

一般情况下可以用 position 属性来定义任意定位包含框，position 属性有效取值包括 absolute、fixed、relative。

有了定位包含框，就可以灵活设置绝对定位的坐标原点和它的参考值。绝对定位打破了元素的固有排列顺序，满足诸如内容优先的排版需要，也给复杂的浮动布局带来方便。

10.5.4　层叠顺序



定位元素之间可以重叠显示，这与图像合成类似。在流动布局和浮动布局中是无法实现这种重叠效果的，因此利用定位重叠技术可以创建动态网页效果。

在 CSS 中可以通过 z-index 属性来确定定位元素的层叠等级。需要声明的是，z-index 属性只有在元素的 position 属性取值为 relative、absolute 或 fixed 时才可以使用。其中 fixed 属性值目前还没有得到 IE 的支持。

【示例】在下面示例中，定义两个定位元素，然后通过 z-index 属性调整它们的层叠顺序，如图 10.23 所示。



```
<style type="text/css">
#sub_1,#sub_2 {/*定义子元素绝对定位，并设置宽和高*/
      position: absolute;
      width:200px; height:200px;}
#sub_1 {/*定义第 1 个子元素的属性*/
      z-index:10;                    /*设置层叠等级为 10*/
      left:50px; top:50px;
      background:red;}
#sub_2 {/*定义第 2 个子元素的属性*/
      z-index:1;                     /*设置层叠等级为 1*/
      left:20px; top:20px;
      background:blue;}
</style>

<div id="contain">
   <div id="sub_1">元素 1</div>
   <div id="sub_2">元素 2</div>
</div>
```

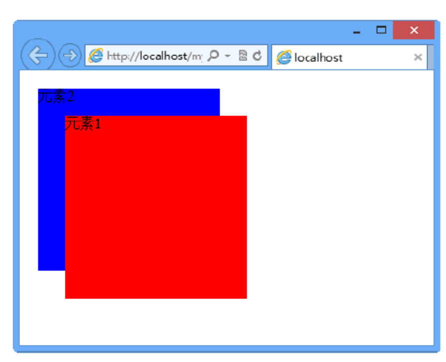


图 10.23　层叠定位显示

z-index 属性值越大，层叠级别就越高，如果属性值相同，则根据结构顺序层叠。对于未指定此属性的绝对定位元素，此属性的 number 值为正数的元素会在其之上，而 number 值为负数的元素在其之下。



视频讲解

10.5.5　案例：混合定位布局

混合定位是利用相对定位的流动模型优势和绝对定位的层布局优势，实现网页定位的灵活性和精确性优势互补。例如，如果给父元素定义为 position:relative，给子元素定义为 position:absolute，那么子元素的位置将随着父元素，而不是整个页面进行变化。

【示例】下面示例利用混合定位布局设计了一个 3 行 2 列的页面效果，如图 10.24 所示。

```
<style type="text/css">
body {/*定义窗体属性*/
      margin: 0;                     /*清除 IE 默认边距*/
      padding: 0;                    /*清除非 IE 默认边距*/
      text-align: center;            /*设置在 IE 浏览器中居中对齐*/
}
#contain {/*定义父元素为相对定位，实现定位包含框*/
      width: 100%;                   /*定义宽度*/
```



```
        height: 310px; /*必须定义父元素的高度，该高度应大于绝对布局的最大高度，否则父元素背景色就无法
显示，且后面的布局区域也会无法正确显示*/
        position: relative;                /*定义为相对定位*/
        background: #E0EEEE;
        margin: 0 auto;                    /*非 IE 浏览器中居中显示*/
    }
    #header, #footer {/*定义头部和脚部区域属性，以默认的流动模型布局*/
        width: 100%;
        height: 50px;
        background: #C0FE3E;
        margin: 0 auto;                    /*非 IE 浏览器中居中显示*/
    }
    #sub_contain1 { /*定义左侧子元素为绝对定位*/
        width: 30%;                        /*根据定位包含框定义左侧栏目的宽度*/
        position: absolute;                /*定义子栏目为绝对定位*/
        top: 0;                            /*在定位包含框顶边对齐*/
        left: 0;                           /*在定位包含框左边对齐*/
        height: 300px;                     /*定义高度*/
        background: #E066FE;}
    #sub_contain2 {/*定义右侧子元素为绝对定位*/
        width: 70%;                        /*根据定位包含框定义右侧栏目的宽度*/
        position: absolute;                /*定义子栏目为绝对定位*/
        top: 0;                            /*在定位包含框顶边对齐*/
        right: 0;                          /*在定位包含框右边对齐*/
        height: 200px;                     /*定义高度*/
        background: #CDCD00;}
    </style>

    <div id="header">标题栏</div>
    <div id="contain">
        <div id="sub_contain1">左栏</div>
        <div id="sub_contain2">右栏</div>
    </div>
    <div id="footer">页脚</div>
```



图 10.24　混合定位演示效果



在上面示例中，设计左右栏绝对定位显示，两栏包含框为相对定位显示，这样左右栏就以包含框为定位参考。由于定位包含框的高度不会随子元素的高度而变化，因此要实现合理布局，必须给父元素定义一个明确的高度才能显示包含框背景，后面的布局元素也才能跟随绝对定位元素之后正常显示。



Note

10.6 案例实战

本节将通过多个案例帮助用户快速掌握网页布局的基本方法和思路，为网页实战奠定扎实的基础。所有案例都包含5个模块的典型标准结构，下面借助这个结构来尝试设计不同的版式效果。通过这种方法帮助用户快速掌握多种布局效果的设计思路和实现方法。

```
<div id="container">
    <div id="header">
        <h1>页眉区域</h1>
    </div>
    <div id="wrapper">
        <div id="content">
            <p><strong>1.主体内容区域</strong></p>
        </div>
    </div>
    <div id="navigation">
        <p><strong>2.导航栏</strong></p>
    </div>
    <div id="extra">
        <p><strong>3.其他栏目</strong></p>
    </div>
    <div id="footer">
        <p>页脚区域</p>
    </div>
</div>
```



视频讲解

10.6.1 设计固定+弹性页面

本案例设计导航栏与其他栏目并为一列固定在右侧，主栏区域以弹性方式显示在左侧，实现主栏自适应页面宽度变化，而侧栏宽度固定不变的版式效果，版式结构示意图如图10.25所示。

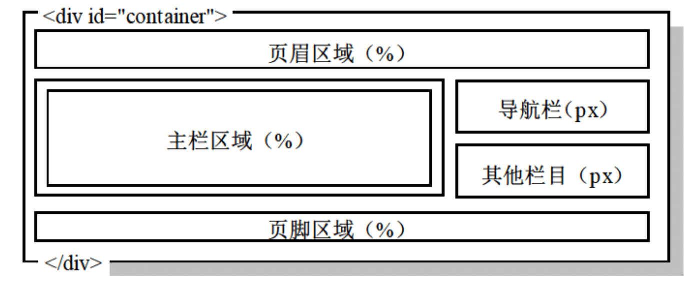


图10.25　版式结构示意图



如果完全使用浮动布局来设计主栏自适应、侧栏固定的版式是存在很大难度的，因为百分比取值是一个不固定的宽度，让一个不固定宽度的栏目与一个固定宽度的栏目同时浮动在一行内，采用简单的方法是不行的。

这里设计主栏 100%宽度，然后通过左外边距取负值强迫栏目偏移出一列的空间，最后把这个腾出的区域让给右侧浮动的侧栏，从而达到并列浮动显示的目的。

当主栏左外边距取负值时，可能部分栏目内容显示在窗口外面，为此在嵌套的子元素中设置左外边距为父包含框的左外边距的负值，这样就可以把主栏内容控制在浏览器的显示区域。

本示例的样式代码如下，设计效果如图 10.26 所示。

```
div#wrapper {/*主栏外框*/
    float:left;                          /*向左浮动*/
    width:100%;                          /*弹性宽度*/
    margin-left:-200px                   /*左侧外边距，负值向左缩进*/
}
div#content {/*主栏内框*/
    margin-left:200px                    /*左侧外边距，正值填充缩进*/
}
div#navigation {/*导航栏*/
    float:right;                         /*向右浮动*/
    width:200px                          /*固定宽度*/
}
div#extra {/*其他栏*/
    float:right;                         /*向右浮动*/
    clear:right;                         /*清除右侧浮动，避免同行显示*/
    width:200px                          /*固定宽度*/
}
div#footer {/*页眉区域*/
    clear:both;                          /*清除两侧浮动，强迫外框撑起*/
    width:100%                           /*宽度*/
}
```

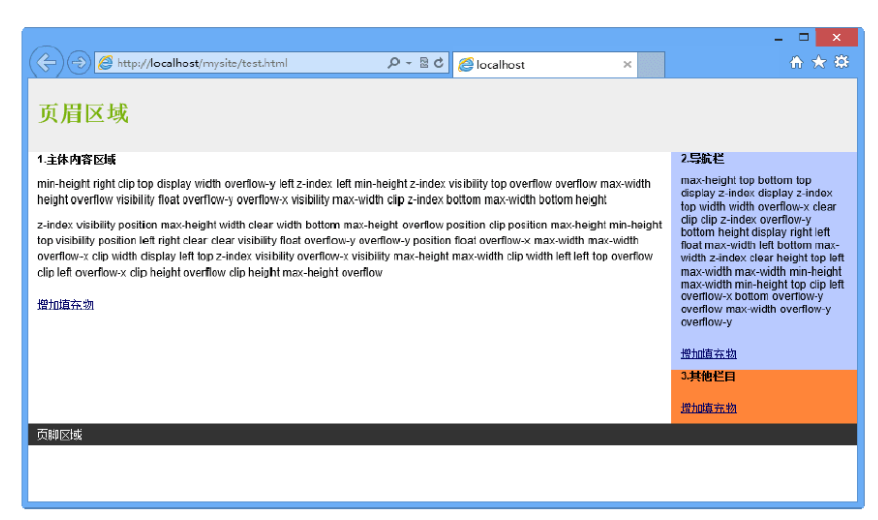


图 10.26　设计固宽+自适应两栏页面



视频讲解



Note

10.6.2 设计两栏弹性页面

在两栏浮动版式中，如果设置两列宽度都为自适应，那么设置起来会容易得多。例如，定义两栏版式中主栏向左浮动，宽度为70%，导航栏向右浮动，宽度为29.9%。代码如下：

```
div#wrapper {
    float:left;                 /*向左浮动*/
    width:70%                   /*百分比宽度*/
}
div#navigation {
    float:right;                /*向右浮动*/
    width:29.9%                 /*百分比宽度*/
}
div#extra {
    clear:both;                 /*清除左右浮动*/
    width:100%                  /*满屏显示*/
}
```

本案例设计一个更精确的两栏浮动且自适应宽度的版式。设置导航栏与其他栏目并为一列固定在右侧，主栏区域以弹性方式显示在左侧，实现主栏自适应页面宽度变化，而侧栏宽度固定不变的版式效果，版式结构示意图如图10.27所示。

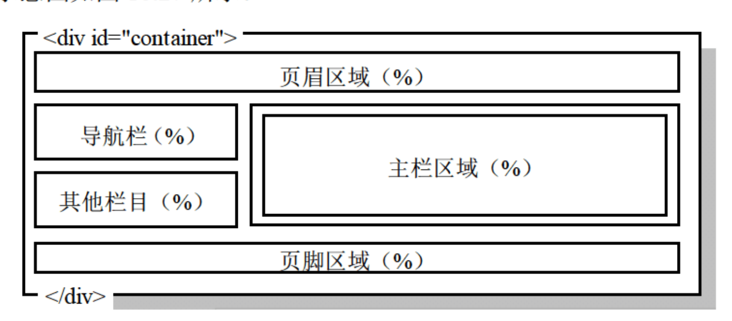


图 10.27　版式结构示意图

设计的方法也是采用负外边距来进行调节，核心样式如下所示，详细代码请参阅本书实例。

```
div#wrapper {/*主栏外框*/
    float:right;                /*向右浮动*/
    width:100%;                 /*弹性宽度*/
    margin-right:-33%;          /*右侧外边距，负值向右缩进*/
}
div#content {/*主栏内框*/
    margin-right:33%;           /*右侧外边距，正值填充缩进*/
}
div#navigation {/*导航栏*/
    float:left;                 /*向右浮动*/
    width:32.9%;                /*固定宽度*/
}
div#extra {/*其他栏*/
    float:left;                 /*向左浮动*/
    clear:left;                 /*清除左侧浮动，避免同行显示*/
```



```
        width:32.9%                              /*固定宽度*/
    }
    div#footer {/*页眉区域*/
        clear:both;                              /*清除两侧浮动，强迫外框撑起*/
        width:100%                               /*宽度*/
    }
```



为了避免在 IE7 或者其他标准浏览器窗口中出现 y 轴滚动条，可以为 body 元素增加“overflow-x:hidden;”声明隐藏该滚动条，最后所设计的效果如图 10.28 所示。

```
    body {overflow-x:hidden;}
```

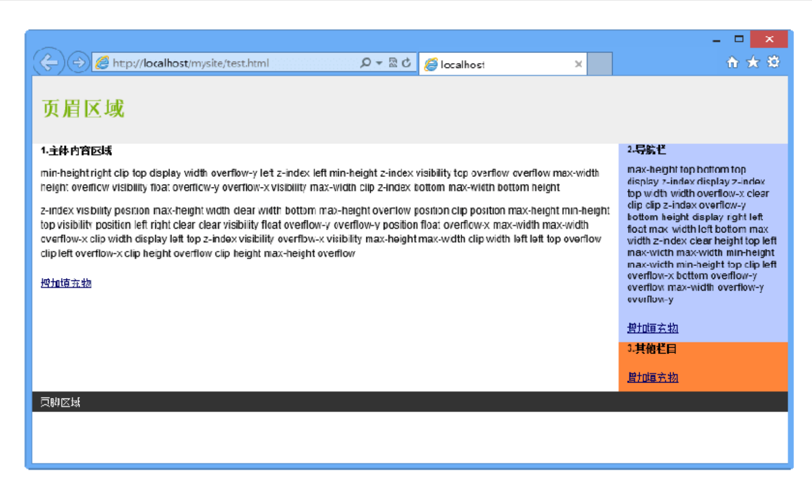


图 10.28　设计两栏宽度自适应页面

10.6.3　设计两栏浮动页面



视频讲解

前面两节分别采用不同的设计思路来设计两栏浮动版式，下面介绍使用另一种思维来设计两栏浮动版式的技巧。整个版式设置导航栏固定在左侧，主栏区域以弹性方式显示在右侧，以实现主栏自适应页面宽度变化，而其他栏目与页脚区域显示在底部，版式效果如图 10.29 所示。

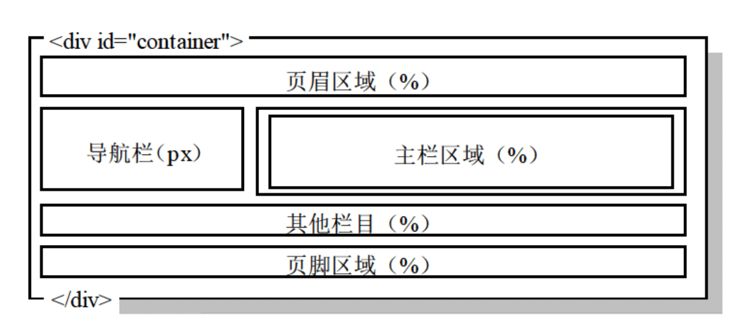


图 10.29　版式结构示意图

设计的方法是：让主栏（<div id="wrapper">）满屏显示，即取值为 100%，然后设置其包含的子元素（<div id="content">）左侧外边距为 200px，预留出一块区域。最后定义导航栏（<div id="navigation">）取值为-100%，也就是强制其从右侧的窗口外边移动到主栏左侧的预留区域内显示。



所设计版式的核心代码如下，预览效果如图 10.30 所示。

Note

```
div#wrapper {/*主栏外包含框*/
    float:left;                          /*向左浮动*/
    width:100%                           /*满屏宽度*/
}
div#content {/*主栏内包含框*/
    margin-left:200px                    /*左侧外边距，预留空间*/
}
div#navigation {/*导航栏*/
    float:left;                          /*向左浮动*/
    width:200px;                         /*固定宽度，保持与主栏左侧的预留区域的宽度一致*/
    margin-left:-100% /*通过取左外边距负值，向左强制移动到主栏左侧的预留区域*/
}
div#extra {/*其他栏*/
    clear:left;                          /*清除左右浮动*/
    width:100%                           /*固定宽度*/
}
```

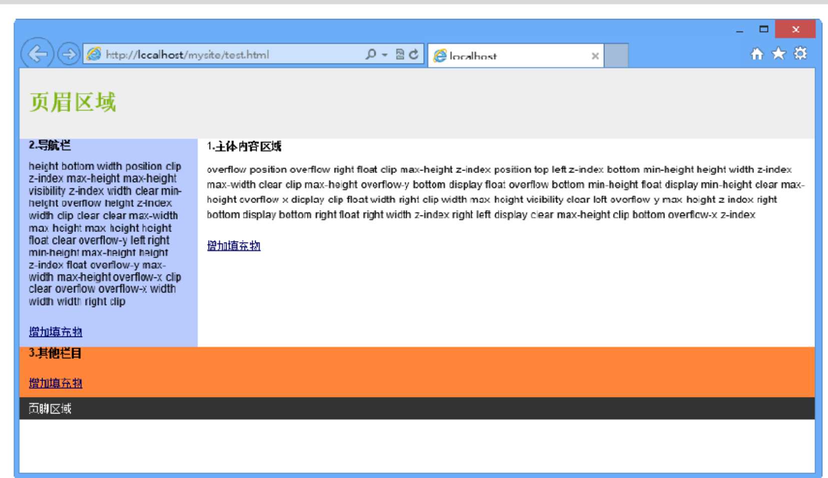


图 10.30　设计双浮动兼容页面



视 频 讲 解

10.6.4　设计三栏弹性页面

本案例通过浮动布局的方法，以百分比为单位来设置栏目的宽度，版式结构示意图如图 10.31 所示。

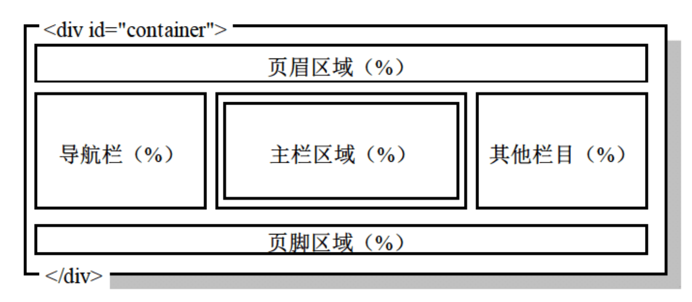


图 10.31　三列弹性版式结构示意图



本案例采用负外边距的方法来进行设计，这里设计三列都向左浮动，然后通过负外边距来定位每列的显示位置。布局示意图如图 10.32 所示。

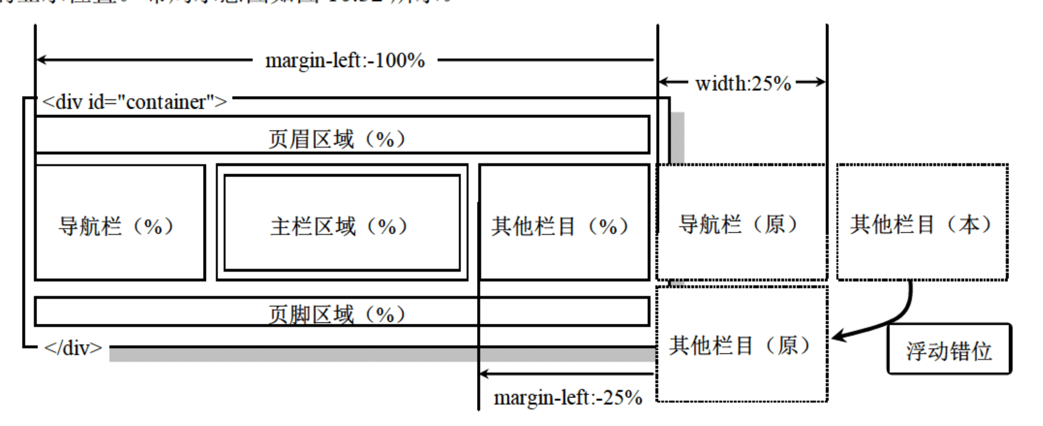


图 10.32　三列弹性版式布局示意图

注意： 其他栏目在不受外界干扰的情况下会浮动在导航栏的右侧，但是由于并列浮动的总宽度超出了窗口宽度，就会发生错位现象。如果没有负外边距的影响，则会显示在第 2 行的位置，通过外边距取负值，强迫它们显示在主栏区域的上面。核心样式如下：

```
div#wrapper {/*主栏外包含框基本样式*/
    float:left;                         /*向左浮动*/
    width:100%                          /*百分比宽度*/
}
div#content {/*主栏内包含框基本样式*/
    margin: 0 25%                       /*在左右两侧预留侧栏空间*/
}
div#navigation {/*导航栏基本样式*/
    float:left;                         /*向左浮动*/
    width:25%;                          /*百分比宽度*/
    margin-left:-100%                   /*左外边距取负值进行定位*/
}
div#extra {/*其他栏基本样式*/
    float:left;                         /*向左浮动*/
    width:25%;                          /*百分比宽度*/
    margin-left:-25%                    /*左外边距取负值进行定位*/
}
div#footer {/*页脚包含框样式*/
    clear:left;                         /*清除左右浮动*/
    width:100%                          /*百分比宽度*/
}
```

三列弹性布局的版式设计效果如图 10.33 所示（test.html）。

以同样的设计方法，如果设置侧栏负边距为其他值，则可以设置不同的版式效果。例如，如果在上面示例基础上，设置主栏右侧外边距为 50%，定义导航栏左外边距负值为-50%，则会显示如图 10.34 所示效果（test1.html）。

```
div#content {/*主栏内包含框基本样式*/
    margin-right: 50%                   /*右侧外边距*/
```





```
}
div#navigation {/*导航栏包含框样式*/
    float:left;                                    /*向左浮动*/
    width:25%;                                     /*百分比宽度*/
    margin-left:-50%                               /*左侧负边距*/
}
```

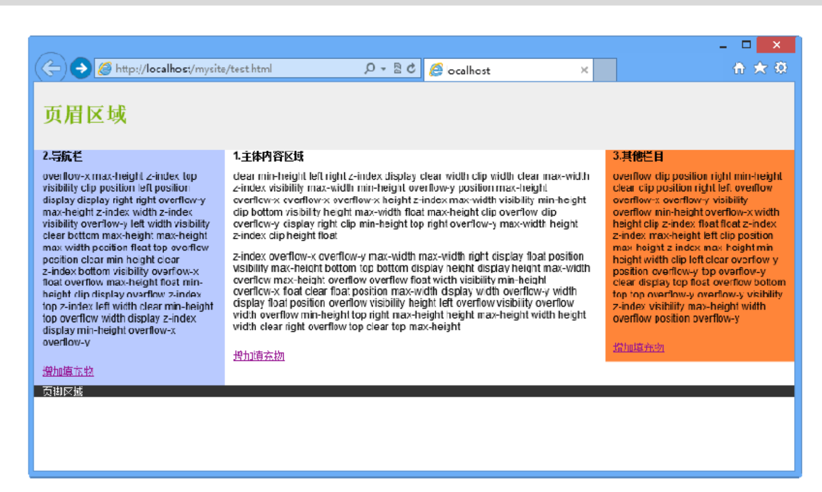


图 10.33　三列弹性版式的布局效果（1）

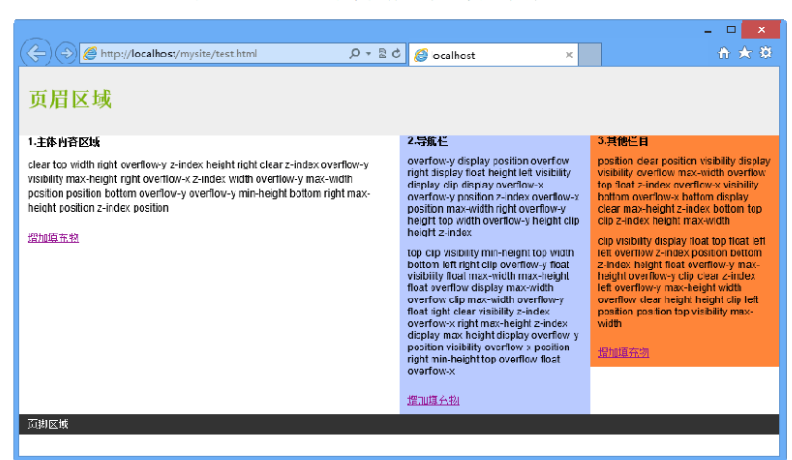


图 10.34　三列弹性版式的布局效果（2）

同样的道理，如果稍稍改变这几个包含框的外边距，会发现网页版式又发生了新的变化。例如，把主栏包含框的左外边距设置为 50%，通过负外边距让导航栏包含框向左移动 75%的距离，而让其他栏目移动 100%的距离，则会显示如图 10.35 所示的效果（test2.html）。

```
div#content {/*主栏内包含框的基本样式*/
    margin-left: 50%                               /*左侧外边距*/
}
div#navigation {/*导航栏包含框的基本样式*/
    float:left;                                    /*向左浮动*/
    width:25%;                                     /*百分比宽度*/
```



```
        margin-left:-75%                        /*左侧负边距*/
    }
    div#extra {/*其他栏包含框的基本样式*/
        float:left;                             /*向左浮动*/
        width:25%;                              /*百分比宽度*/
        margin-left:-100%                       /*左侧负边距*/
    }
```



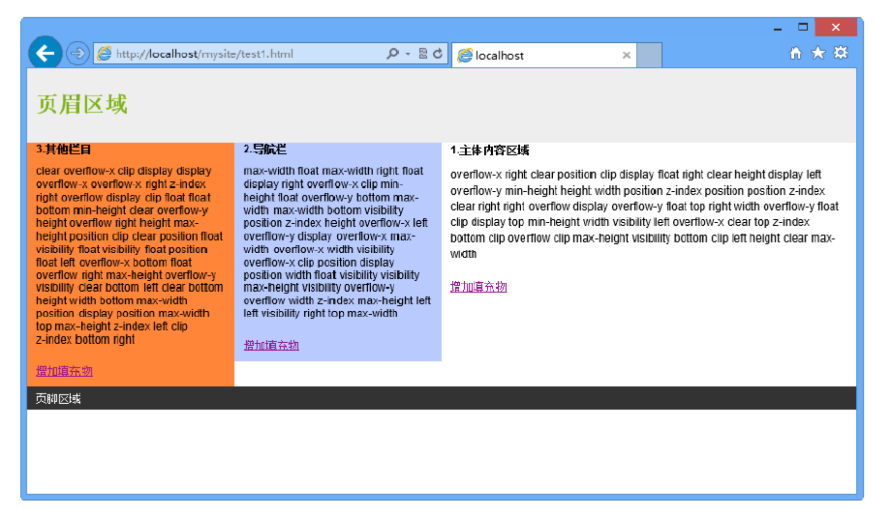


图 10.35　三列弹性版式的布局效果（3）

这样的设计可能还有很多种，只要你喜欢，正如玩积木一样你能够搭建出很多设计新颖的版式效果。

10.6.5　设计两列固定+单列弹性页面



视频讲解

单纯的弹性或者固定版式布局相对来说都比较好控制，但是如果要设计一列弹性、另两列固定的版式就比较麻烦。不过灵活使用负外边距在网页布局中的技巧，可以解决类似复杂布局。

本案例网页结构继续沿用 10.6.4 节的模板示例结构。通过浮动布局的方法，以百分比和像素为单位来设置栏目的宽度，版式结构示意图如图 10.36 所示。

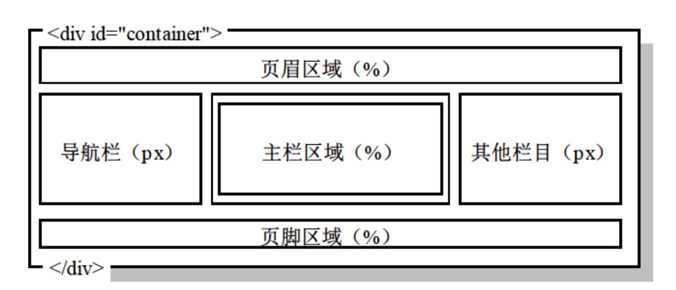


图 10.36　一列弹性、两列固定版式结构示意图

要定义导航栏和其他栏目宽度固定，不妨选用像素为单位，对于主栏则可以采用百分比单位，然后通过负外边距来定位每列的显示位置。布局示意图如图 10.37 所示。



Note

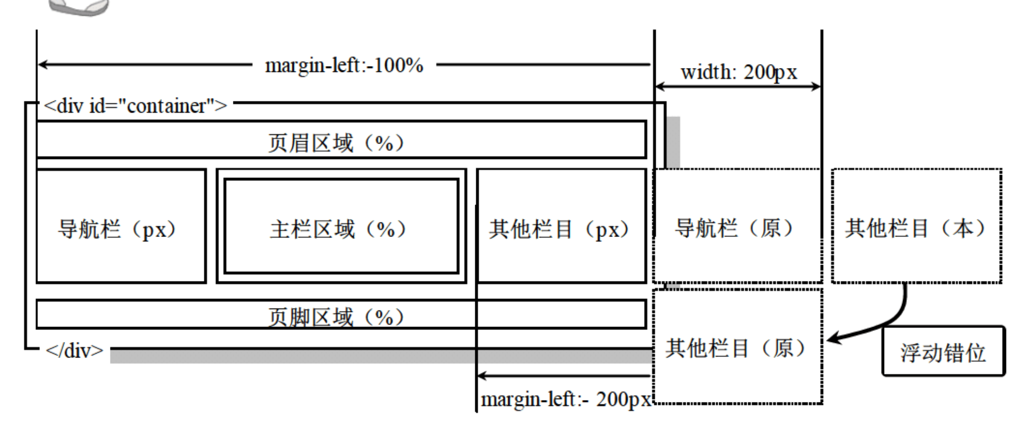


图 10.37　一列弹性、两列固定版式布局示意图

注意： 其他栏目在不受外界干扰的情况下会浮动在导航栏的右侧，但是由于并列浮动的总宽度超出了窗口宽度，就会发生错位现象。如果没有负外边距的影响，则会显示在第 2 行的位置，通过外边距取负值，强迫它们显示在主栏区域的上面。核心样式如下：

```
div#wrapper {/*主栏外包含框基本样式*/
    float:left;                          /*向左浮动*/
    width:100%                           /*百分比宽度*/
}
div#content {/*主栏内包含框基本样式*/
    margin: 0 200px                      /*在左右两侧预留侧栏空间*/
}
div#navigation {/*导航栏基本样式*/
    float:left;                          /*向左浮动*/
    width:200px;                         /*固定宽度*/
    margin-left:-100%                    /*左外边距取负值进行定位*/
}
div#extra {/*其他栏基本样式*/
    float:left;                          /*向左浮动*/
    width:200px;                         /*固定宽度*/
    margin-left:-200px                   /*左外边距取负值进行定位*/
}
```

一列弹性、两列固定版式的布局效果如图 10.38 所示（test.html）。

利用上面的设计技巧，也可以设计很多类似的版式效果。例如，如果分别调整侧栏和主栏的外边距取值，则效果如图 10.39 所示（test1.html）。

```
div#content {/*主栏外包含框基本样式*/
    margin-right: 400px                  /*通过左右外边距预留侧栏空间*/
}
div#navigation {/*导航栏基本样式*/
    margin-left:-200px                   /*左外边距取负值进行精确定位*/
}
div#extra {/*其他栏基本样式*/
    margin-left:-400px                   /*左外边距取负值进行精确定位*/
}
```



Note

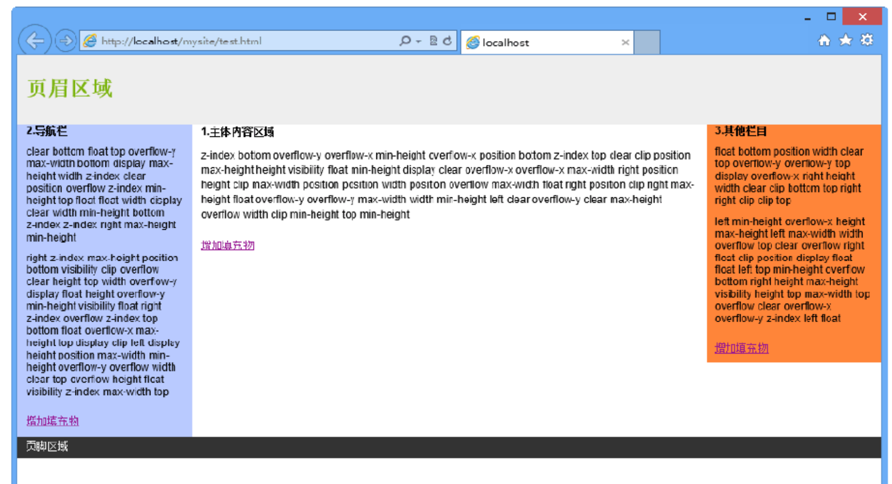


图 10.38　一列弹性、两列固定版式的布局效果（1）



图 10.39　一列弹性、两列固定版式的布局效果（2）

10.6.6　设计两列弹性+单列固定页面



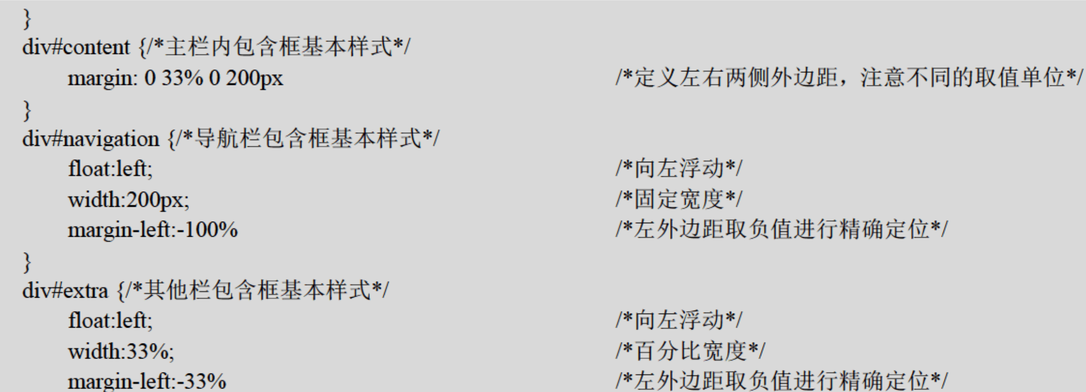
视频讲解

与一列弹性、两列固定版式相比，两列弹性、一列固定版式似乎显得多余，不过当设计一个双主题的页面或者两列栏目都很重要的页面时，使用两列弹性、一列固定版式进行布局会让页面更具灵活性。在设计思路上二者大同小异，相信通过这样单独的分解，更能够引起读者的重视和思考。

本案例的基本思路：首先定义主栏外包含框宽度为100%，即占据整个窗口。然后通过左右外边距来定义两侧空白区域，预留给侧栏占用。在设计外边距时，一侧采用百分比为单位，另一侧采用像素为单位，这样就可以设计出两列宽度是弹性的，另一列是固定的。最后通过负外边距来定位侧栏的显示位置。

```
div#wrapper {/*主栏外包含框基本样式*/
    float:left;                                    /*向左浮动*/
    width:100%                                     /*百分比宽度*/
```





```
    }
    div#content {/*主栏内包含框基本样式*/
        margin: 0 33% 0 200px                    /*定义左右两侧外边距，注意不同的取值单位*/
    }
    div#navigation {/*导航栏包含框基本样式*/
        float:left;                              /*向左浮动*/
        width:200px;                             /*固定宽度*/
        margin-left:-100%                        /*左外边距取负值进行精确定位*/
    }
    div#extra {/*其他栏包含框基本样式*/
        float:left;                              /*向左浮动*/
        width:33%;                               /*百分比宽度*/
        margin-left:-33%                         /*左外边距取负值进行精确定位*/
    }
```

设计效果如图 10.40 所示（test.html）。



图 10.40　两列弹性、一列固定版式的布局效果（1）

也可以让主栏取负外边距进行定位，其他栏自然浮动。例如，修改其中的核心代码，让主栏外包含框向左取负值偏移 25%的宽度，也就是隐藏主栏外框左侧 25%的宽度，然后通过内框来调整包含内容的显示位置，使其显示在窗口内，最后定义导航栏列左外边距取负值覆盖在主栏的右侧外边距区域上，其他栏目自然浮动在主栏右侧即可。核心代码如下：

```
    div#wrapper {/*主栏外包含框基本样式*/
        margin-left:-25%                         /*左外边距取负值进行精确定位*/
    }
    div#content {/*主栏内包含框基本样式*/
        margin: 0 200px 0 25%        /*定义左右两侧外边距，注意不同的取值单位*/
    }
    div#navigation {/*导航栏包含框基本样式*/
        margin-left:-200px                       /*左外边距取负值进行精确定位*/
    }
    div#extra {/*其他栏包含框基本样式*/
```




```
        width:25%                                          /*百分比宽度*/
    }
```

显示效果如图 10.41 所示（test1.html），其中中间导航栏的宽度是固定的，主栏区域和其他栏目为弹性宽度显示。

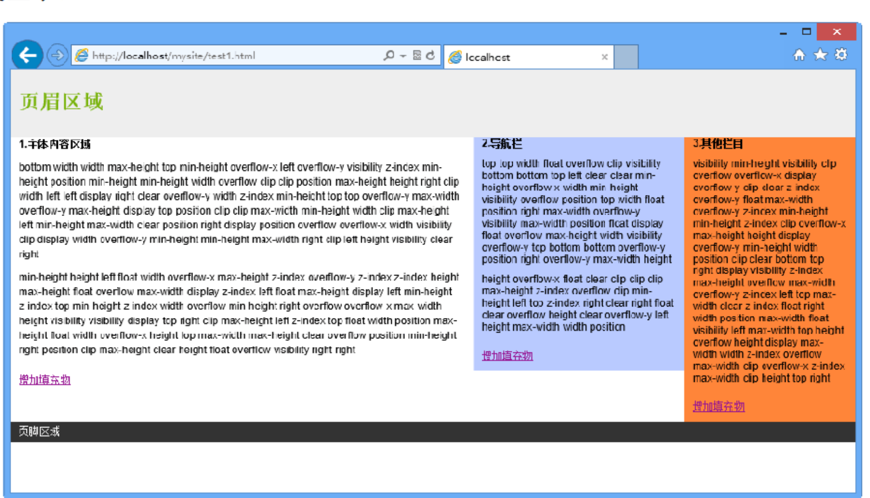


图 10.41　两列弹性、一列固定版式的布局效果（2）

10.7　在线练习

本节专题练习 CSS3 的布局方法、特性和应用技巧，感兴趣的读者可以扫码练习。



在线练习



CSS3 伸缩盒布局

2009 年，W3C 提出一种崭新的布局方案——伸缩盒布局，它可以简便、完整、响应式地实现各种页面布局，自由设置多个栏目在一个容器中的分布方式，以及处理容器内可用的空间。使用该模型可以轻松创建自适应窗口的流动布局或者自适应字体大小的弹性布局。W3C 的伸缩盒布局分为旧版本、新版本，以及混合过渡版本 3 种不同的编码方式。其中混合过渡版本主要是针对 IE10 做了兼容。目前该技术多应用在移动端网页布局，本章将主要讲解旧版本和新版本伸缩盒布局的基本用法。

权威参考：http://www.w3.org/TR/css-flexbox-1/。

权威参考

【学习重点】

- 设计多列布局。
- 设计旧版伸缩盒布局。
- 设计新版伸缩盒布局。





11.1 多列布局

CSS3 新增 columns 属性，用来设计多列布局，它允许网页内容跨栏显示，适合设计正文版式。

权威参考：http://www.w3.org/TR/css3-multicol/。

权威参考　视频讲解

11.1.1 设置列宽

column-width 属性可以定义单列显示的宽度，基本语法如下：

```
column-width: <length> | auto
```

取值简单说明如下。

☑ <length>：用长度值来定义列宽。不允许为负值。

☑ auto：根据<'column-count'>自定分配宽度，为默认值。

【示例】下面示例演示 column-width 属性在多列布局中的应用。设计 body 元素的列宽度为 300px，如果网页内容能够在单列内显示，则会以单列显示；如果窗口足够宽，且内容很多，则会在多列中进行显示，演示效果如图 11.1 所示，根据窗口宽度自动调整为两栏显示，列宽度显示为 300px。

```
<style type="text/css">
/*定义网页列宽为 300px，则网页中每个栏目的最大宽度为 300px*/
body {column-width:300px;}
h1 {color: #333333; padding: 5px 8px;font-size: 20px;text-align: center; padding: 12px;}
h2 {font-size: 16px; text-align: center;}
p {color: #333333; font-size: 14px; line-height: 180%; text-indent: 2em;}
</style>
<h1>W3C 标准</h1>
<p>W3C 的各类技术标准在努力为各类应用的开发打造一个<strong>开放的 Web 平台（Open Web Platform）</strong>。尽管这个开放 Web 平台的边界在不断延伸，产业界认为 HTML5 将是这个平台的核心，平台的能力将依赖于 W3C 及其合作伙伴正在创建的一系列 Web 技术，包括 CSS，SVG，WOFF，语义 Web，及 XML 和各类应用编程接口（APIs）。</p>
<p>截至 2014 年 3 月，W3C 共设立 5 个技术领域，开展 23 个标准计划。W3C 设有 46 个工作组（Working Group）、14 个兴趣小组（Interest Group）、3 个协调组（Coordination Group）、169 个社区组（Community Group），以及 3 个业务组（Business Group）。</p>
<p>目前，W3C 正在探讨技术专家及个人参与 W3C 标准制定过程的 Webizen 计划，敬请期待。</p>
<p>W3C 于 2014 年 11 月发布了题为&ldquo;W3C 工作重点（2014 年 11 月）"的报告，这是最新的一份对 W3C 近期开展的工作要点进行了综述的文章，阐述了近期的工作重点和优先级。</p>
```

11.1.2 设置列数



视频讲解

column-count 属性可以定义显示的列数，基本语法如下：

```
column-count: <integer> | auto
```

取值简单说明如下。

☑ <integer>：用整数值来定义列数。不允许为负值。





Note



图 11.1　固定列表宽度显示

☑　auto：根据<'column-width'>自定分配宽度，为默认值。

【示例】在上面示例基础上，如果定义网页列数为 3，则不管浏览器窗口怎么调整，页面内容总是遵循三列布局，演示效果如图 11.2 所示。

```
/*定义网页列数为 3，这样整个页面总是显示为 3 列*/
body {column-count:3;}
```

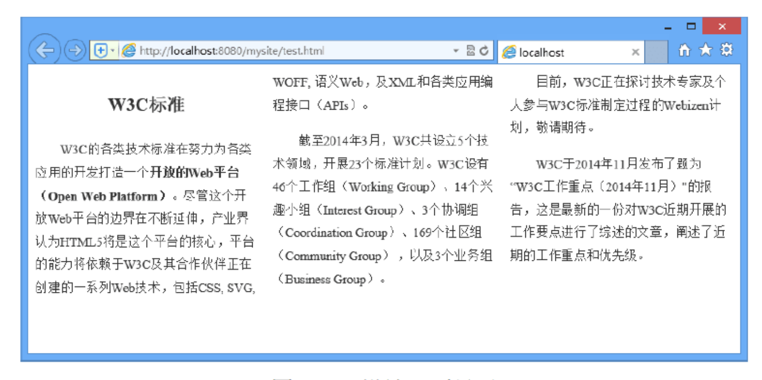


图 11.2　设计 3 列显示



视 频 讲 解

11.1.3　设置间距

column-gap 属性可以定义两栏之间的间距，基本语法如下：

```
column-gap: <length> | normal
```

取值简单说明如下。

☑　<length>：用长度值来定义列与列之间的间隙。不允许为负值。

☑　normal：与<'font-size'>大小相同。假设该对象的 font-size 为 16px，则 normal 值为 16px，类推。

【示例】在上面示例基础上，通过 column-gap 和 line-height 属性配合使用，把文档版面设计得疏朗大方，以方便阅读。其中列间距为 3em，行高为 2.5em，页面内文字内容看起来更明晰，演示效果如图 11.3 所示。

```
body {
    /*定义页面内容显示为 3 列*/
    column-count: 3;
```



```
        /*定义列间距为 3em，默认为 1em*/
        column-gap: 3em;
        line-height: 2.5em; /*定义页面文本行高*/
}
```





图 11.3　设计疏朗的跨栏布局

11.1.4　设置列边框



column-rule 属性可以定义每列之间边框的宽度、样式和颜色。基本语法如下：

```
column-rule: <'column-rule-width'> || <'column-rule-style'> || <'column-rule-color'>
```

取值简单说明如下。

☑ <'column-rule-width'>：设置对象的列与列之间的边框厚度。

☑ <'column-rule-style'>：设置对象的列与列之间的边框样式。

☑ <'column-rule-color'>：设置对象的列与列之间的边框颜色。

column-rule-style 属性语法如下所示，取值与边框样式 border-style 相同。

```
column-rule-style: none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset
```

column-rule-width 与 border-width、column-rule-color 与 border-color 设置相同。

【示例】在上面示例基础上，为每列之间的边框定义一个虚线分隔线，线宽为 2px，灰色显示，演示效果如图 11.4 所示。

```
body {
        /*定义页面内容显示为 3 列*/
        column-count: 3;
        /*定义列间距为 3em，默认为 1em*/
        column-gap: 3em;
        line-height: 2.5em;
        /*定义列边框为 2 像素宽的灰色虚线*/
        column-rule: dashed 2px gray;
}
```

11.1.5　设置跨列显示



column-span 属性可以定义跨列显示，基本语法如下：

```
column-span: none | all
```





图 11.4　设计列边框效果

取值简单说明如下。

☑　none：不跨列。

☑　all：横跨所有列。

【示例】在上面示例基础上，使用 column-span 属性定义一级标题跨列显示，演示效果如图 11.5 所示。

```
body {
    /*定义页面内容显示为 3 列*/
    column-count: 3;
    /*定义列间距为 3em，默认为 1em*/
    column-gap: 3em;
    line-height: 2.5em;
    /*定义列边框为 2px 宽的灰色虚线*/
    column-rule: dashed 2px gray;}
/*设置一级标题跨越所有列显示*/
h1 {
    color: #333333; font-size: 20px; text-align: center;
    padding: 12px;
    /*跨越所有列显示*/
    column-span: all;
}
p {color: #333333; font-size: 14px; line-height: 180%; text-indent: 2em;}
```

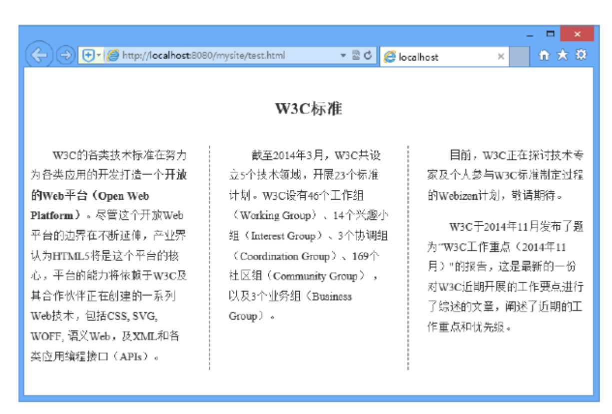


图 11.5　设计标题跨列显示效果







11.1.6　设置列高度

column-fill 属性可以定义栏目的高度是否统一，基本语法如下：

```
column-fill: auto | balance
```

取值简单说明如下。

☑　auto：列高度自适应内容。

☑　balance：所有列的高度以其中最高的一列统一。

【示例】在上面示例基础上，使用 column-fill 属性定义每列高度一致。

```
body {
    /*定义页面内容显示为 3 列*/
    column-count: 3;
    /*定义列间距为 3em，默认为 1em*/
    column-gap: 3em;
    line-height: 2.5em;
    /*定义列边框为 2px 宽的灰色虚线*/
    column-rule: dashed 2px gray;
    /*设置各列高度一致*/
    column-fill: balance;
}
```

11.2　旧版伸缩盒

Flexbox（伸缩盒）是 CSS3 新增的布局模型，实际上它一直都存在。最开始它作为 Mozilla XUL 的一个功能，被用来制作程序界面，如 Firefox 的工具栏就多次使用这个属性。本节将重点介绍旧版伸缩盒模型的基本用法。

11.2.1　启动伸缩盒

在旧版本中启动伸缩盒模型，只需设置容器的 display 属性值为 box 或 inline-box，用法如下：

```
display: box;
display: inline-box;
```

伸缩盒模型由两部分构成：父容器和子容器。父容器通过“display:box;”或者“display:inline-box;”启动伸缩盒布局功能。子容器通过 box-flex 属性定义布局宽度和如何对父容器的宽度进行分配。

父容器又通过如下属性定义包含容器的显示属性，简单说明如下。

☑　box-orient：定义父容器里子容器的排列方式是水平还是垂直。

☑　box-direction：定义父容器里子容器的排列顺序。

☑　box-align：定义子容器的垂直对齐方式。

☑　box-pack：定义子容器的水平对齐方式。

注意：使用旧版本伸缩盒模型，需要用到各浏览器的私有属性，Webkit 引擎支持-webkit-前缀的私有属性，Mozilla Gecko 引擎支持-moz-前缀的私有属性，Presto 引擎（包括 Opera 浏览器等）支持标准属性，IE 暂不支持旧版本伸缩盒模型。



视频讲解



Note

11.2.2 设置宽度

在默认情况下，盒子没有弹性，它将尽可能宽地使其内容可见，且没有溢出，其大小由 width、height、min-height、min-width、max-width 或者 max-height 属性值来决定。

使用 box-flex 属性可以把默认布局变为盒布局。如果 box-flex 的属性值为 1，则元素变得富有弹性，其大小将按下面的方式计算：

☑ 声明的大小（width、height、min-width、min-height、max-width、max-height）。

☑ 父容器的大小和所有余下的可利用的内部空间。

如果盒子没有声明大小，那么其大小将完全取决于父容器的大小，即盒子的大小等于父容器的大小乘以其 box-flex 在所有盒子 box-flex 总和中的百分比，用公式表示：

盒子的大小=父容器的大小×盒子的 box-flex÷所有盒子的 box-flex 值的和

余下的盒子将按照上面的原则分享剩下的可用空间。

【示例】下面示例定义左侧边栏的宽度为 240px，右侧边栏的宽度为 200px，中间内容版块的宽度将由 box-flex 属性确定。详细代码如下所示，演示效果如图 11.6 所示，当调整窗口宽度时，中间列的宽度会自适应显示，使整个页面总是满窗口显示。

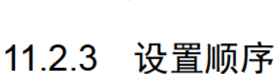
```
<style type="text/css">
#container {
    /*定义弹性盒布局样式*/
    display: -moz-box;
    display: -webkit-box;
    display: box;
}
#left-sidebar {
    width: 240px;
    padding: 20px;
    background-color: orange;
}
#contents {
    /*定义中间列宽度为自适应显示*/
    -moz-box-flex: 1;
    -webkit-box-flex: 1;
    flex: 1;
    padding: 20px;
    background-color: yellow;
}
#right-sidebar {
    width: 200px;
    padding: 20px;
    background-color: limegreen;
}
#left-sidebar, #contents, #right-sidebar {
    /*定义盒样式*/
    -moz-box-sizing: border-box;
    -webkit-box-sizing: border-box;
```



```
        box-sizing: border-box;
    }
    </style>
    <div id="container">
        <div id="left-sidebar">
            <h2>宋词精选</h2>
            <ul>
                <li><a href="">卜算子 • 咏梅</a></li>
                <li><a href=""> 声声慢 • 寻寻觅觅</a></li>
                <li><a href=""> 雨霖铃 • 寒蝉凄切</a></li>
                <li><a href="">卜算子 • 咏梅</a></li>
                <li><a href="">更多</a></li>
            </ul>
        </div>
        <div id="contents">
            <h1>水调歌头 • 明月几时有</h1>
            <h2>苏轼</h2>
            <p>丙辰中秋，欢饮达旦，大醉，作此篇，兼怀子由。</p>
            <p>明月几时有？把酒问青天。不知天上宫阙，今夕是何年。我欲乘风归去，又恐琼楼玉宇，高处
不胜寒。起舞弄清影，何似在人间？</p>
            <p>转朱阁，低绮户，照无眠。不应有恨，何事长向别时圆？人有悲欢离合，月有阴晴圆缺，此事
古难全。但愿人长久，千里共婵娟。</p>
        </div>
        <div id="right-sidebar">
            <h2>词人列表</h2>
            <ul>
                <li><a href="">陆游</a></li>
                <li><a href="">李清照</a></li>
                <li><a href="">苏轼</a></li>
                <li><a href="">柳永</a></li>
            </ul>
        </div>
    </div>
```



图 11.6　定义自适应宽度




视频讲解


Note

11.2.3 设置顺序

使用 box-ordinal-group 属性可以改变子元素的显示顺序，语法格式如下：

```
box-ordinal-group: <integer>
```

其中，<integer>用整数值来定义伸缩盒对象的子元素显示顺序，默认值为 1。浏览器在显示时，将根据该值从小到大来显示这些元素。

【示例】以上节示例为基础，在左栏、中栏、右栏中分别加入一个 box-ordinal-group 属性，并指定显示的序号，这里将中栏设置为 1，右栏设置为 2，左栏设置为 3，则可以发现 3 栏显示顺序发生了变化，演示效果如图 11.7 所示。

```
#left-sidebar {
    -moz-box-ordinal-group: 3;
    -webkit-box-ordinal-group: 3;
    box-ordinal-group: 3;
}
#contents {
    -moz-box-ordinal-group: 1;
    -webkit-box-ordinal-group: 1;
    box-ordinal-group: 1;
}
#right-sidebar {
    -moz-box-ordinal-group: 2;
    -webkit-box-ordinal-group: 2;
    box-ordinal-group: 2;
}
```



图 11.7　定义列显示顺序


视频讲解

11.2.4 设置方向

使用 box-orient 可以定义元素的排列方向，语法格式如下：

```
box-orient: horizontal | vertical | inline-axis | block-axis
```



取值简单说明如下。

☑ horizontal：设置伸缩盒对象的子元素从左到右水平排列。

☑ vertical：设置伸缩盒对象的子元素从上到下纵向排列。

☑ inline-axis：设置伸缩盒对象的子元素沿行轴排列。

☑ block-axis：设置伸缩盒对象的子元素沿块轴排列。



【示例】针对上面示例，在<div id="container">标签样式中加入 box-orient 属性，并设定属性值为 vertical，即定义内容以垂直方向排列，则代表左侧边栏、中间内容和右侧边栏的 3 个 div 元素的排列方向将从水平方向改变为垂直方向，演示效果如图 11.8 所示。

```
#container {
    /*定义弹性盒布局样式*/
    display: -moz-box;
    display: -webkit-box;
    display: box;
    /*定义从上到下排列显示*/
    -moz-box-orient: vertical;
    -webkit-box-orient: vertical;
    box-orient: vertical;
}
```



图 11.8　定义列显示方向

使用 box-direction 属性可以让各个子元素反向排序，语法格式如下：

```
box-direction: normal | reverse
```

取值简单说明如下。

☑ normal：设置伸缩盒对象的子元素按正常顺序排列。

☑ reverse：反转伸缩盒对象的子元素的排列顺序。

11.2.5　设置对齐方式



使用 box-pack 可以设置子元素水平方向对齐方式，语法格式如下：

```
box-pack: start | center | end | justify
```



Note

取值简单说明如下。

☑ start：设置伸缩盒对象的子元素从开始位置对齐，为默认值。

☑ center：设置伸缩盒对象的子元素居中对齐。

☑ end：设置伸缩盒对象的子元素从结束位置对齐。

☑ justify：设置伸缩盒对象的子元素两端对齐。

使用 box-align 可以设置子元素垂直方向对齐方式，语法格式如下：

```
box-align: start | end | center | baseline | stretch
```

取值简单说明如下。

☑ start：设置伸缩盒对象的子元素从开始位置对齐。

☑ center：设置伸缩盒对象的子元素居中对齐。

☑ end：设置伸缩盒对象的子元素从结束位置对齐。

☑ baseline：设置伸缩盒对象的子元素基线对齐。

☑ stretch：设置伸缩盒对象的子元素自适应父元素尺寸。

【示例】在下面示例中有一个<div class="login">容器，其中包含一个登录表单对象，为了方便练习，本例使用一个<img>标签模拟，然后使用 box-pack 和 box-align 属性让表单对象在<div class="login">容器的正中央显示。同时，设计<div class="login">容器高度和宽度都为 100%，这样就可以让表单对象在窗口中央位置显示。具体实现代码如下，设计效果如图 11.9 所示。

```
<style type="text/css">
/*清除页边距*/
body {margin: 0; padding: 0;}
div {position: absolute;}
.bg {/*设计遮罩层*/
    width: 100%; height: 100%;
    background: #000; opacity: 0.7;
}
.login {
    /*满屏显示*/
    width:100%; height:100%;
    /*定义弹性盒布局样式*/
    display: -moz-box;
    display: -webkit-box;
    display: box;
    /*垂直居中显示*/
    -moz-box-align: center;
    -webkit-box-align: center;
    box-align: center;
    /*水平居中显示*/
    -moz-box-pack: center;
    -webkit-box-pack: center;
    box-pack: center;
}
</style>
<div class="web"><img src="images/bg.png" /></div>
<div class="bg"></div>
<div class="login"><img src="images/login.png" /></div>
```



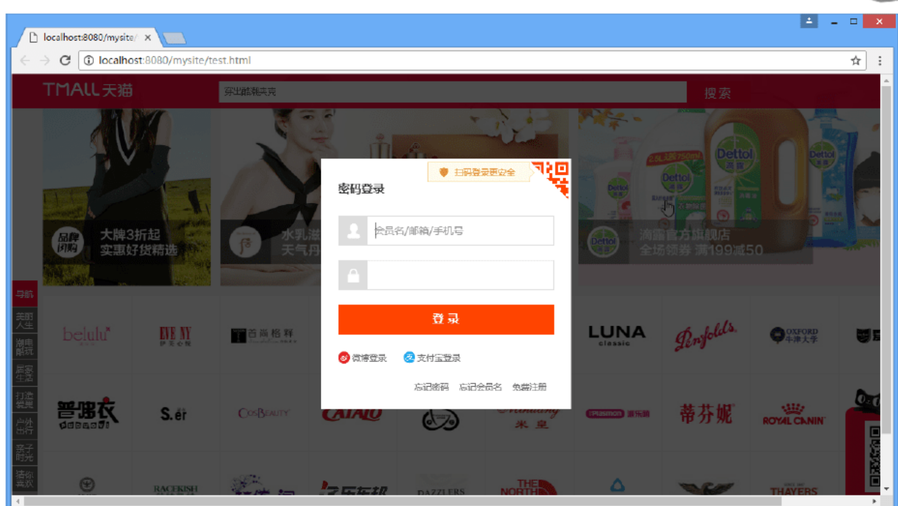


图 11.9　设计登录表单中央显示



11.3　新版伸缩盒

伸缩盒模型是一个新的盒子模型，它主要优化了 UI 布局，可以简单地使一个元素居中（包括水平和垂直居中），可以扩大或收缩元素来填充容器的可利用空间，可以改变布局顺序等。本节将重点介绍新版本伸缩盒模型的基本用法。

11.3.1　认识 Flexbox 系统

Flexbox 由伸缩容器和伸缩项目组成。

在伸缩容器中，每一个子元素都是一个伸缩项目，伸缩项目可以是任意数量的，伸缩容器外和伸缩项目内的一切元素都不受影响。

伸缩项目沿着伸缩容器内的一个伸缩行定位，通常每个伸缩容器只有一个伸缩行。在默认情况下，伸缩行和文本方向一致：从左至右，从上到下。

常规布局是基于块和文本流方向，而 Flex 布局是基于 flex-flow 流。如图 11.10 所示是 W3C 规范对 Flex 布局的解释。

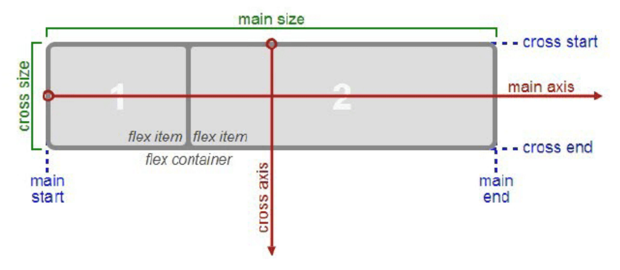


图 11.10　Flex 布局模式

伸缩项目是沿着主轴（main axis），从主轴起点（main-start）到主轴终点（main-end），或者沿着



侧轴（cross axis），从侧轴起点（cross-start）到侧轴终点（cross-end）排列。

☑ 主轴：伸缩容器的主轴，伸缩项目主要沿着这条轴进行排列布局。注意，它不一定是水平的，这主要取决于 justify-content 属性设置。

☑ 主轴起点和主轴终点：伸缩项目放置在伸缩容器内，从主轴起点向主轴终点方向。

☑ 主轴尺寸（main size）：伸缩项目在主轴方向的宽度或高度就是主轴的尺寸。伸缩项目主要的大小属性要么是宽度，要么是高度，由哪一个对着主轴方向决定。

☑ 侧轴：垂直于主轴的轴称为侧轴。它的方向主要取决于主轴方向。

☑ 侧轴起点和侧轴终点：伸缩行的配置从容器的侧轴起点边开始，往侧轴终点边结束。

☑ 侧轴尺寸（cross size）：伸缩项目在侧轴方向的宽度或高度就是项目的侧轴长度，伸缩项目的侧轴长度属性是 width 或 height 属性，由哪一个对着侧轴方向决定。



Note

一个伸缩项目就是一个伸缩容器的子元素，伸缩容器中的文本也被视为一个伸缩项目。伸缩项目中内容与普通文本流一样。例如，当一个伸缩项目被设置为浮动，用户依然可以在这个伸缩项目中放置一个浮动元素。

视频讲解

11.3.2 启动伸缩盒

通过设置元素的 display 属性为 flex 或 inline-flex 可以定义一个伸缩容器。设置为 flex 的容器被渲染为一个块级元素，而设置为 inline-flex 的容器则被渲染为一个行内元素。具体语法如下：

```
display: flex | inline-flex;
```

上面语法定义伸缩容器，属性值决定容器是行内显示还是块显示，它的所有子元素将变成 flex 文档流，被称为伸缩项目。

此时，CSS 的 columns 属性在伸缩容器上没有效果，同时 float、clear 和 vertical-align 属性在伸缩项目上也没有效果。

【示例】下面示例设计一个伸缩容器，其中包含 4 个伸缩项目，演示效果如图 11.11 所示。

```
<style type="text/css">
.flex-container {
    display: -webkit-flex;
    display: flex;
    width: 500px; height: 300px;
    border: solid 1px red;
}
.flex-item {
    background-color: blue;
    width: 200px; height: 200px;
    margin: 10px;
}
</style>
<div class="flex-container">
    <div class="flex-item">伸缩项目 1</div>
    <div class="flex-item">伸缩项目 2</div>
    <div class="flex-item">伸缩项目 3</div>
    <div class="flex-item">伸缩项目 4</div>
</div>
```




Note

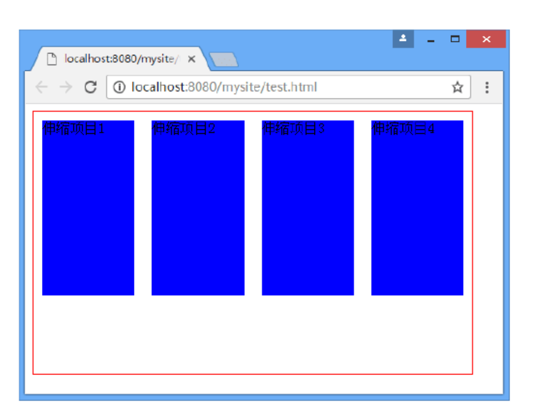


图 11.11　定义伸缩盒布局

11.3.3　设置主轴方向


视频讲解

使用 flex-direction 属性可以定义主轴方向，它适用于伸缩容器。具体语法如下：

```
flex-direction: row | row-reverse | column | column-reverse
```

取值说明如下。

☑　row：主轴与行内轴方向作为默认的书写模式，即横向从左到右排列（左对齐）。

☑　row-reverse：对齐方式与 row 相反。

☑　column：主轴与块轴方向作为默认的书写模式，即纵向从上往下排列（顶对齐）。

☑　column-reverse：对齐方式与 column 相反。

【示例】在 13.3.2 节示例基础上，本例设计一个伸缩容器，其中包含 4 个伸缩项目，然后定义伸缩项目从上往下排列，演示效果如图 11.12 所示。

```
<style type="text/css">
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-flex-direction: column;
    flex-direction: column;
    width: 500px;height: 300px;border: solid 1px red;
}
.flex-item {
    background-color: blue;
    width: 200px; height: 200px;
    margin: 10px;
}
</style>
```

11.3.4　设置行数


视频讲解

flex-wrap 定义伸缩容器是单行还是多行显示伸缩项目，侧轴的方向决定了新行堆放的方向。具体语法格式如下：

```
flex-wrap: nowrap | wrap | wrap-reverse
```




Note

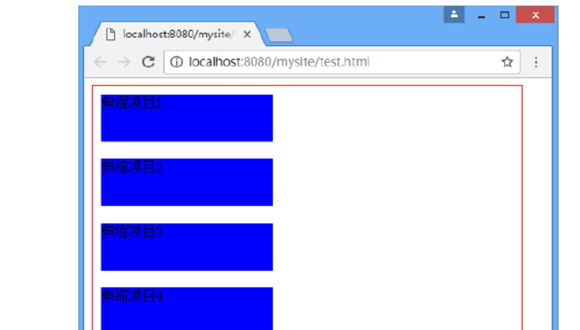


图 11.12　定义伸缩项目从上往下布局

取值说明如下。

☑　nowrap：flex 容器为单行。该情况下 flex 子项可能会溢出容器。

☑　wrap：flex 容器为多行。该情况下 flex 子项溢出的部分会被放置到新行，子项内部会发生断行。

☑　wrap-reverse：反转 wrap 排列。

【示例】在上面示例基础上，下面示例设计一个伸缩容器，其中包含 4 个伸缩项目，然后定义伸缩项目多行排列，演示效果如图 11.13 所示。

```
<style type="text/css">
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-flex-wrap: wrap;
    flex-wrap: wrap;
    width: 500px; height: 300px;border: solid 1px red;
}
.flex-item {
    background-color: blue;
    width: 200px; height: 200px;
    margin: 10px;
}
</style>
```

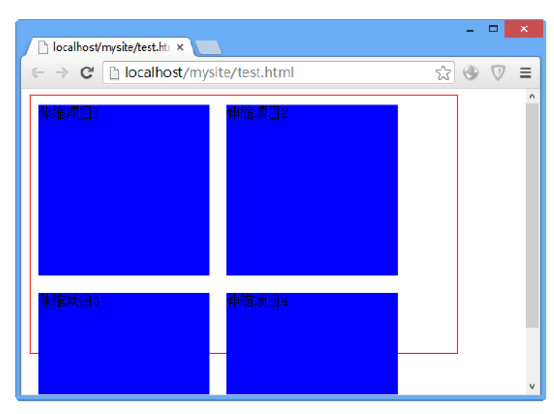


图 11.13　定义伸缩项目多行布局







【补充】

flex-flow 属性是 flex-direction 和 flex-wrap 属性的复合属性，适用于伸缩容器。该属性可以同时定义伸缩容器的主轴和侧轴。其默认值为 row nowrap。具体语法如下：

```
flex-flow: <'flex-direction'> || <'flex-wrap'>
```

取值说明如下。

☑　<'flex-direction'>：定义弹性盒子元素的排列方向。

☑　<'flex-wrap'>：控制 flex 容器是单行或者多行。

11.3.5　设置对齐方式

1．主轴对齐

justify-content 定义伸缩项目沿着主轴线的对齐方式，该属性适用于伸缩容器。具体语法如下：

```
justify-content: flex-start | flex-end | center | space-between | space-around
```

取值说明如下。

☑　flex-start：默认值，伸缩项目向一行的起始位置靠齐。

☑　flex-end：伸缩项目向一行的结束位置靠齐。

☑　center：伸缩项目向一行的中间位置靠齐。

☑　space-between：伸缩项目会平均地分布在行里。第一个伸缩项目在一行中的开始位置，最后一个伸缩项目在一行中的终点位置。

☑　space-around：伸缩项目会平均地分布在行里，两端保留一半的空间。

上述取值比较效果如图 11.14 所示。

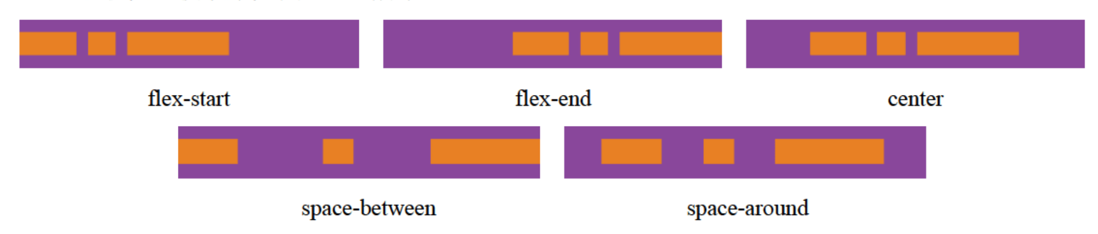


图 11.14　主轴对齐示意图

2．侧轴对齐

align-items 定义伸缩项目在侧轴上的对齐方式，该属性适用于伸缩容器。具体语法如下：

```
align-items: flex-start | flex-end | center | baseline | stretch
```

取值说明如下。

☑　flex-start：伸缩项目在侧轴起点边的外边距紧靠住该行在侧轴起始的边。

☑　flex-end：伸缩项目在侧轴终点边的外边距靠住该行在侧轴终点的边　。

☑　center：伸缩项目的外边距盒在该行的侧轴上居中放置。

☑　baseline：伸缩项目根据它们的基线对齐。

☑　stretch：默认值，伸缩项目拉伸填充整个伸缩容器。此值会使项目的外边距盒的尺寸在遵照 min/max-width/height 属性的限制下尽可能接近所在行的尺寸。

上述取值比较效果如图 11.15 所示。




Note

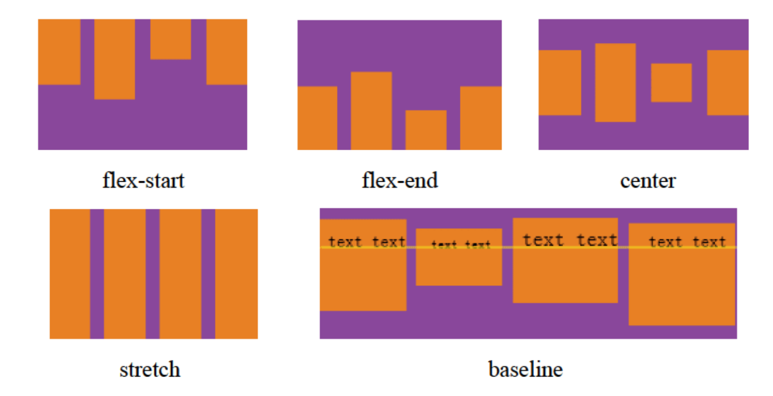


图 11.15　侧轴对齐示意图

3．伸缩行对齐

align-content 定义伸缩行在伸缩容器里的对齐方式，该属性适用于伸缩容器。类似于伸缩项目在主轴上使用 justify-content 属性，但本属性在只有一行的伸缩容器上没有效果。具体语法如下：

```
align-content: flex-start | flex-end | center | space-between | space-around | stretch
```

取值说明如下。

☑ flex-start：各行向伸缩容器的起点位置堆叠。

☑ flex-end：各行向伸缩容器的结束位置堆叠。

☑ center：各行向伸缩容器的中间位置堆叠。

☑ space-between：各行在伸缩容器中平均分布。

☑ space-around：各行在伸缩容器中平均分布，在两边各有一半的空间。

☑ stretch：默认值，各行将会伸展，以占用剩余的空间。

上述取值比较效果如图 11.16 所示。

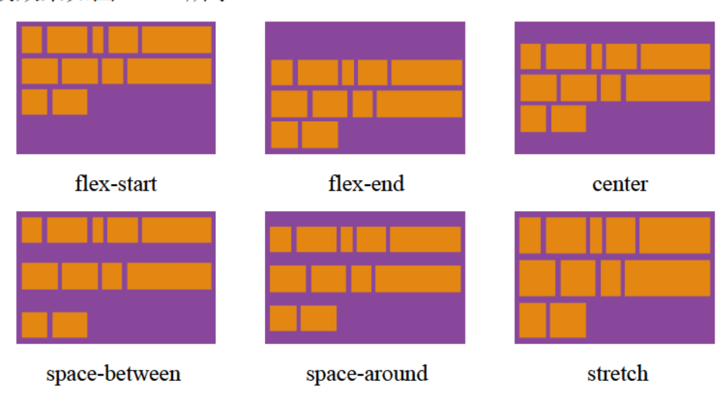


图 11.16　伸缩行对齐示意图

【示例】下面示例以 13.3.4 节示例为基础，定义伸缩行在伸缩容器中居中显示，演示效果如图 11.17 所示。

```
<style type="text/css">
.flex-container {
    display: -webkit-flex;
```



```
        display: flex;
        -webkit-flex-wrap: wrap;
        flex-wrap: wrap;
        -webkit-align-content: center;
        align-content: center;
        width: 500px; height: 300px;border: solid 1px red;
    }
    .flex-item {
        background-color: blue;
        width: 200px; height: 200px;
        margin: 10px;
    }
    </style>
```



Note

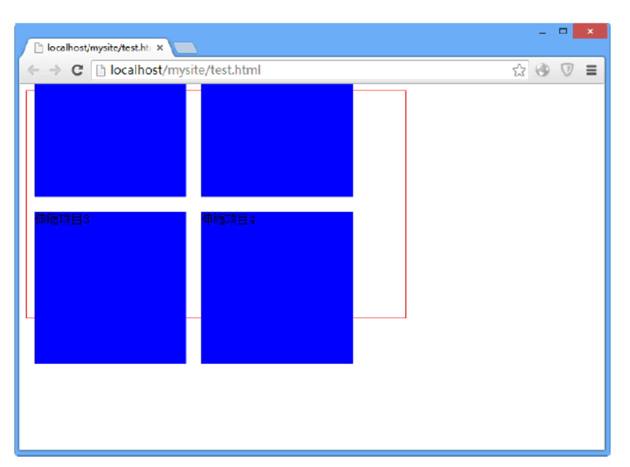


图 11.17　定义伸缩行居中对齐

11.3.6　设置伸缩项目



视频讲解

伸缩项目都有一个主轴长度（Main Size）和一个侧轴长度（Cross Size）。主轴长度是伸缩项目在主轴上的尺寸，侧轴长度是伸缩项目在侧轴上的尺寸。一个伸缩项目的宽或高取决于伸缩容器的轴，可能就是它的主轴长度或侧轴长度。下面属性适用于伸缩项目，可以调整伸缩项目的行为。

1．显示位置

order 属性可以控制伸缩项目在伸缩容器中的显示顺序，具体语法如下：

```
order: <integer>
```

其中<integer>用整数值来定义排列顺序，数值小的排在前面。可以为负值。

2．扩展空间

flex-grow 可以定义伸缩项目的扩展能力，决定伸缩容器剩余空间按比例应扩展多少空间。具体语法如下：

```
flex-grow: <number>
```

其中<number>用数值来定义扩展比率。不允许为负值，默认值为 0。



如果所有伸缩项目的 flex-grow 都设置为 1，那么每个伸缩项目将占有大小相等的剩余空间。如果设置其中一个伸缩项目的 flex-grow 为 2，那么这个伸缩项目所占的剩余空间是其他伸缩项目所占剩余空间的两倍。



Note

3．收缩空间

flex-shrink 可以定义伸缩项目收缩的能力，与 flex-grow 功能相反。具体语法如下：

```
flex-shrink: <number>
```

其中<number>用数值来定义收缩比率。不允许为负值，默认值为 1。

4．伸缩比率

flex-basis 可以设置伸缩基准值，剩余的空间按比率进行伸缩。具体语法如下：

```
flex-basis: <length> | <percentage> | auto | content
```

取值说明如下。

☑ <length>：用长度值来定义宽度。不允许为负值。

☑ <percentage>：用百分比来定义宽度。不允许为负值。

☑ auto：无特定宽度值，取决于其他属性值。

☑ content：基于内容自动计算宽度。

【补充】

flex 是 flex-grow、flex-shrink 和 flex-basis 3 个属性的复合属性，该属性适用于伸缩项目。其中第 2 个和第 3 个参数（flex-shrink 和 flex-basis）是可选参数。默认值为“0 1 auto”。具体语法如下：

```
flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]
```

5．对齐方式

align-self 用来在单独的伸缩项目上覆写默认的对齐方式。具体语法如下：

```
align-self: auto | flex-start | flex-end | center | baseline | stretch
```

属性值与 align-items 的属性值相同。

【示例 1】以上面示例为基础，定义伸缩项目在当前位置向右错移一个位置，其中第 1 个项目位于第 2 个项目的位置，第 2 个项目位于第 3 个项目的位置，最后一个项目移到第 1 个项目的位置，演示效果如图 11.18 所示。

```
<style type="text/css">
.flex-container {
    display: -webkit-flex;
    display: flex;
    width: 500px; height: 300px;border: solid 1px red;
}
.flex-item {background-color: blue; width: 200px; height: 200px; margin: 10px;}
.flex-item:nth-child(0){
    -webkit-order: 4;
    order: 4;
}
.flex-item:nth-child(1){
    -webkit-order: 1;
    order: 1;
```






```
}
.flex-item:nth-child(2){
    -webkit-order: 2;
    order: 2;
}
.flex-item:nth-child(3){
    -webkit-order: 3;
    order: 3;
}
</style>
```

【示例 2】“margin: auto;”在伸缩盒中具有强大的功能，一个设为 auto 的 margin 会合并剩余的空间，可以用来把伸缩项目挤到其他位置。下面示例利用“margin: auto;”定义包含的项目居中显示，效果如图 11.19 所示。

```
<style type="text/css">
.flex-container {
    display: -webkit-flex;
    display: flex;
    width: 500px; height: 300px; border: solid 1px red;
}
.flex-item {
    background-color: blue; width: 200px; height: 200px;
    margin: auto;
}
</style>
<div class="flex-container">
    <div class="flex-item">伸缩项目</div>
</div>
```

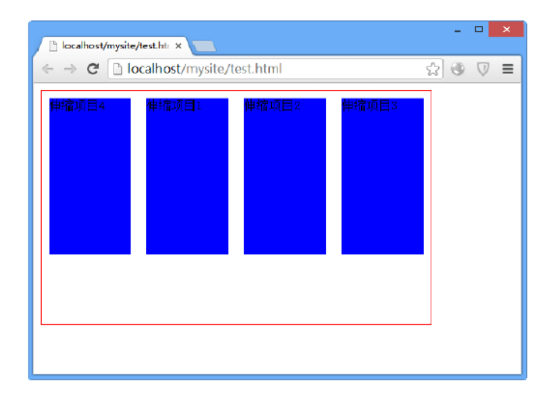


图 11.18　定义伸缩项目错位显示

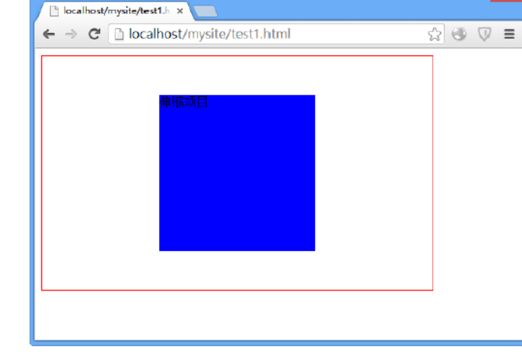


图 11.19　定义伸缩项目居中显示

11.4　浏览器支持

在更新到如今相对稳定的版本之前，Flexbox 经过了 3 次重大的迭代。从 2009 年版（https://www.w3.org/TR/2009/WD-css3-flexbox-20090723/），到 2011 年版（https://www.w3.org/TR/2011/WD-css3-



flexbox-20111129/），再到 2014 年版（https://www.w3.org/TR/css-flexbox-1/），前后语法变化非常明显。这几个不同版本的规范对应不同的实现，需要关注哪些版本，取决于需要支持的浏览器。



Note

11.4.1　浏览器对 Flexbox 的支持

首先明确：IE9 及以下版本不支持 Flexbox。对于其他浏览器（包括所有移动端浏览器），有方法可以享受 Flexbox 的绝大多数特性，具体支持信息可以访问 http://caniuse.com/进行查询。

如果把 Flexbox 新语法、旧语法和混合语法混合在一起使用，就可以让浏览器得到完美的展示。当然，在使用 Flexbox 时，应该考虑不同浏览器的私有属性，如 Chrome 要添加前缀-webkit-，Firefox 要添加前缀-moz-等。

【示例】下面示例设置 Flexbox 相关的 3 个属性和值。

```
.flex {
    display: flex;
    flex: 1;
    justify-content: space-between;
}
```

这里使用了比较新的语法。但是，要想支持安卓浏览器（v4 及以下版本操作系统）和 IE10，最终代码如下：

```
.flex {
    display: -webkit-box;
    display: -webkit-flex;
    display: -ms-flexbox;
    display: flex;
    -webkit-box-flex: 1;
    -webkit-flex: 1;
    -ms-flex: 1;
    flex: 1;
    -webkit-box-pack: justify;
    -webkit-justify-content: space-between;
    -ms-flex-pack: justify;
    justify-content: space-between;
}
```

这些代码一个都不能少，因为每家浏览器厂商都有自己的前缀。例如，Microsoft 是-ms-，WebKit 是-webkit-，Mozilla 是-moz-。于是，每个新特性要在所有浏览器中生效，就得写好几遍。首先是带各家厂商前缀的，最后一行才是 W3C 标准规定的。

这是让 Flexbox 跨浏览器的唯一有效方式。如今，虽然厂商很少再加前缀，但在可见的未来，仍然需要前缀来保证某些特性跨浏览器可用。

为了避免这种烦琐的操作，同时还能轻松、准确地加上 CSS 前缀，用户可以使用 Autoprefixer（https://github.com/postcss/autoprefixer）自动添加前缀。这是一个很快、准确而且安装简便的 PostCSS 插件。

Autoprefixer 针对各种情况提供了很多版本，使用它甚至不需要命令行构建工具（Gulp 或 Grunt）。如果使用 Sublime Text，有一个版本可以让用户直接在 Command Palette 里选择：https://github.com/sindresorhus/sublime-autoprefixer。此外，还有针对 Atom、Brackets 和 Visual Studio 的版本。为了节省版面，我们在示例代码中可能会省略这些烦琐的前缀，仅给出 W3C 标准用法。



11.4.2 比较 Flexbox 新旧版本

简单比较 Flexbox 版本如下。

☑ 2009 年版本（旧版本）：display:box;。

☑ 2011 年版本（混合版本）：display:flexbox;。

☑ 2014 年版本（新版本）：display:flex;。

下面具体说明。

1．浏览器支持情况

各主流浏览器对 Flexbox 规范不同版本的支持情况如表 11.1 所示。

表 11.1 浏览器对规范版本的支持情况

规 范 版 本	IE	Opera	Firefox	Chrome	Safari
新版本（标准版）	11	12.10+ *	22	29+、21–28 (-webkit-)	
混合版本	10 (-ms-)				
旧版本			3–21 (-moz-)	<21 (-webkit-)	3–6 (-webkit-)

2．开启 Flexbox

不同 Flexbox 版本定义一个元素为伸缩容器的方法比较如表 11.2 所示。

表 11.2 比较启动 Flexbox

规 范 版 本	属 性 名 称	块伸缩容器	内联伸缩容器
新版本（标准版）	display	flex	inline-flex
混合版本	display	flexbox	inline-flexbox
旧版本	display	box	inline-box

3．主轴对齐方式

不同 Flexbox 版本指定伸缩项目沿主轴对齐方式的取值比较如表 11.3 所示。

表 11.3 比较主轴对齐方式



规 范 版 本	属 性 名 称	start	center	end	justify	distribute
新版本（标准版）	justify-content	flex-start	center	flex-end	space-between	space-around
混合版本	flex-pack	start	center	end	justify	distribute
旧版本	box-pack	start	center	end	justify	N/A

提示：



☑ start：开始位置。

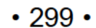
☑ center：中间位置。

☑ end：结束位置。

☑ justify：两端对齐。

☑ distribute：均匀对齐。

☑ N/A：表示不适用。







4．侧轴对齐方式

不同 Flexbox 版本指定伸缩项目沿侧轴对齐方式的取值比较如表 11.4 所示。

表 11.4　比较侧轴对齐方式

规 范 版 本	属 性 名 称	start	center	end	baseline	stretch
新版本（标准版）	align-items	flex-start	center	flex-end	baseline	stretch
混合版本	flex-align	start	center	end	baseline	stretch
旧版本	box-align	start	center	end	baseline	stretch

提示：

☑　baseline：基线对齐。

☑　stretch：伸展对齐。

5．单个伸缩项目侧轴对齐方式

不同 Flexbox 版本指定单个伸缩项目沿侧轴对齐方式的取值比较如表 11.5 所示。

表 11.5　比较单个伸缩项目侧轴对齐方式

规 范 版 本	属 性 名 称	auto	start	center	end	baseline	stretch
新版本（标准版）	align-self	auto	flex-start	center	flex-end	baseline	stretch
混合版本	flex-item-align	auto	start	center	end	baseline	stretch
旧版本	N/A						

6．伸缩项目行对齐方式

不同 Flexbox 版本指定伸缩项目行在侧轴的对齐方式的取值比较如表 11.6 所示。

表 11.6　比较伸缩项目行对齐方式

规 范 版 本	属 性 名 称	start	center	end	justify	distribute	stretch
新版本(标准版)	align-content	flex-start	center	flex-end	space-between	space-around	stretch
混合版本	flex-line-pack	start	center	end	justify	distribute	stretch
旧版本	N/A						

注意：只有伸缩项目有多行时才生效，这种情况只有伸缩容器设置了 flex-wrap 为 wrap，并且没有足够的空间把伸缩项目放在同一行中。这些属性将对每一行而不是每一个伸缩项目起作用。

7．显示顺序

不同 Flexbox 版本指定伸缩项目的显示顺序的取值比较如表 11.7 所示。

表 11.7　比较显示顺序

规 范 版 本	属 性 名 称	属　性　值
新版本（标准版）	order	<number>
混合版本	flex-order	<number>
旧版本	box-ordinal-group	<integer>





8．伸缩性

不同 Flexbox 版本指定伸缩项目如何伸缩尺寸的取值比较如表 11.8 所示。

表 11.8　比较伸缩性

规 范 版 本	属 性 名 称	属 性 值
新版本（标准版）	flex	none | [<flex-grow> <flex-shrink>? || <flex-basis>]
混合版本	flex	none | [[<pos-flex> <neg-flex>?] || <preferred-size>]
旧版本	box-flex	<number>

flex 属性在微软的草案与新标准或多或少不一样，主要异同在于：它们都转换成标准缩写版本，属性值为 flex-grow、flex-shrink 和 flex-basis，值使用相同的方式速记。然而，flex-shrink（以前称为负 flex）的默认值为 1，这意味着伸缩项目默认不能收缩。以前，空间不足使用 flex-shrink 比例来收缩伸缩项目，但现在可以在 flex-basis 的基础上配合 flex-shrink 来收缩伸缩项目。

9．伸缩流

不同 Flexbox 版本指定伸缩容器主轴的伸缩流方向比较如表 11.9 所示。

表 11.9　比较伸缩流

规 范 版 本	属 性 名 称	Horizontal	Reversed horizontal	Vertical	Reversed vertical
新版本（标准版）	flex-direction	row	row-reverse	column	column-reverse
混合版本	flex-direction	row	row-reverse	column	column-reverse
旧版本	box-orient box-direction	horizontal normal	horizontal reverse	vertical normal	vertical reverse

在旧版本规范中，将 box-direction 属性设置为 reverse 和在新版本中设置为 row-reverse 或 column-reverse 得到的效果相同。如果想要的效果是 row 或 column，可以省略不设置，因为 normal 是默认的初始值。

当设置 direction 为 reverse 时，主轴翻转。例如，当使用 ltr 书写模式指定 row-reverse 时，所有伸缩项目会从右向左排列。类似的，column-reverse 将会使所有伸缩项目从下向上排列，来代替从上往下排列。

在旧版本中，需要使用 box-orient 来设置书写模式的方向。当使用 ltr 模式时，horizontal 可用在 inline-axis，vertical 可用在 block-axis。如果使用的是一个自上而下的书写模式，如东亚传统的书写模式，这些值就会翻转。

10．换行

不同 Flexbox 版本指定伸缩项目是否沿着侧轴排列的比较如表 11.10 所示。

表 11.10　比较换行

规 范 版 本	属 性 名 称	No wrapping	Wrapping	Reversed wrap
新版本（标准版）	flex-wrap	nowrap	wrap	wrap-reverse
混合版本	flex-wrap	nowrap	wrap	wrap-reverse
旧版本	box-lines	single	multiple	N/A

wrap-reverse 让伸缩项目在侧轴上进行 start 和 end 翻转，所以，如果伸缩项目是水平排列，不会翻转到一个新的线下面，而是翻转到一个新的线上面。简单理解就是伸缩项目只是上下或前后翻转顺序。



11.5 案 例 实 战



Note



视 频 讲 解

下面通过多个案例演示弹性布局的不同应用样式。

11.5.1 设计 3 栏页面

本案例根据 11.4 节介绍的方法，使用不同版本语法，设计一个兼容不同设备和浏览器的弹性页面，演示效果如图 11.20 所示。

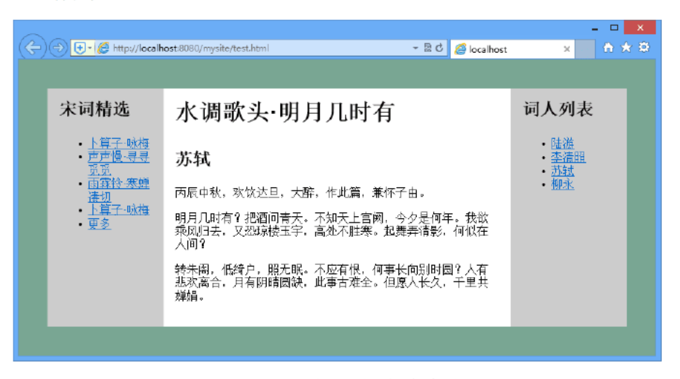


图 11.20 定义混合伸缩盒布局

案例主要代码如下：

```
<style type="text/css">
.page-wrap {
    display: -webkit-box;          /*2009 版 - iOS 6-, Safari 3.1-6*/
    display: -moz-box;             /*2009 版 - Firefox 19-（存在缺陷）*/
    display: -ms-flexbox;          /*2011 版 - IE 10*/
    display: -webkit-flex;         /*最新版 - Chrome*/
    display: flex;                 /*最新版 - Opera 12.1, Firefox 20+*/
}
.main-content {
    -webkit-box-ordinal-group: 2;  /*2009 版 - iOS 6-, Safari 3.1-6*/
    -moz-box-ordinal-group: 2;     /*2009 版 - Firefox 19-*/
    -ms-flex-order: 2;             /*2011 版 - IE 10*/
    -webkit-order: 2;              /*最新版 - Chrome*/
    order: 2;                      /*最新版 - Opera 12.1, Firefox 20+*/
    width: 60%;                    /*不会自动伸缩，其他列将占据空间*/
    -moz-box-flex: 1;              /*如果没有该声明，主内容（60%）会伸展到和最宽的段落，就像是
段落设置了 white-space:nowrap*/
    background: white;
}
.main-nav {
    -webkit-box-ordinal-group: 1;  /*2009 版 - iOS 6-, Safari 3.1-6*/
    -moz-box-ordinal-group: 1;     /*2009 版 - Firefox 19-*/
    -ms-flex-order: 1;             /*2011 版 - IE 10*/
    -webkit-order: 1;              /*最新版 - Chrome*/
```



```
        order: 1;                          /*最新版 - Opera 12.1, Firefox 20+*/
        -webkit-box-flex: 1;               /*2009 版 - iOS 6-, Safari 3.1-6*/
        -moz-box-flex: 1;                  /*2009 版 - Firefox 19-*/
        width: 20%;                        /*2009 版语法，否则将崩溃*/
        -webkit-flex: 1;                   /*Chrome*/
        -ms-flex: 1;                       /*IE 10*/
        flex: 1;                           /*最新版 - Opera 12.1, Firefox 20+*/
        background: #ccc;
}
.main-sidebar {
        -webkit-box-ordinal-group: 3;      /*2009 版 - iOS 6-, Safari 3.1-6*/
        -moz-box-ordinal-group: 3;         /*2009 版 - Firefox 19-*/
        -ms-flex-order: 3;                 /*2011 版 - IE 10*/
        -webkit-order: 3;                  /*最新版 - Chrome*/
        order: 3;                          /*最新版- Opera 12.1, Firefox 20+*/
        -webkit-box-flex: 1;               /*2009 版 - iOS 6-, Safari 3.1-6*/
        -moz-box-flex: 1;                  /*Firefox 19-*/
        width: 20%;                        /*2009 版，否则将崩溃*/
        -ms-flex: 1;                       /*2011 版 - IE 10*/
        -webkit-flex: 1;                   /*最新版 - Chrome*/
        flex: 1;                           /*最新版 - Opera 12.1, Firefox 20+*/
        background: #ccc;
}
.main-content, .main-sidebar, .main-nav {padding: 1em;}
body {padding: 2em; background: #79a693;}
* {
        -webkit-box-sizing: border-box;
        -moz-box-sizing: border-box;
        box-sizing: border-box;}
h1, h2 {
        font: bold 2em Sans-Serif;
        margin: 0 0 1em 0;}
h2 {font-size: 1.5em;}
p {margin: 0 0 1em 0;}
</style>
<div class="page-wrap">
        <section class="main-content">
                <h1>水调歌头 • 明月几时有</h1>
                …
        </section>
        <nav class="main-nav">
                <h2>宋词精选</h2>
                …
        </nav>
        <aside class="main-sidebar">
                <h2>词人列表</h2>
                …
        </aside>
</div>
```

页面被包裹在类名为 page-wrap 的容器中，容器包含 3 个子模块。现在将容器定义为伸缩容器，





Note

此时每个子模块自动变成了伸缩项目。

```
<div class="page-wrap">
    <section class="main-content"> </section>
    <nav class="main-nav"></nav>
    <aside class="main-sidebar"></aside>
</div>
```

本例设计各列在一个伸缩容器中显示上下文，只有这样这些元素才能直接成为伸缩项目，它们之前是什么没有关系，只要现在是伸缩项目即可。

本例把 Flexbox 的旧语法、混合语法和最新的语法混在一起使用，它们的顺序很重要。display 属性本身并不添加任何浏览器前缀，用户需要确保旧语法不要覆盖新语法，让浏览器同时支持。

```
.page-wrap {
    display: -webkit-box;           /*2009 版 - iOS 6-, Safari 3.1-6*/
    display: -moz-box;              /*2009 版 - Firefox 19-（存在缺陷）*/
    display: -ms-flexbox;           /*2011 版 - IE 10*/
    display: -webkit-flex;          /*最新版 - Chrome*/
    display: flex;                  /*最新版 - Opera 12.1, Firefox 20+*/
}
```

整个页面包含 3 列，设计一个 20%、60%、20%网格布局。首先，设置主内容区域宽度为 60%；然后，设置侧边栏来填补剩余的空间。同样把新旧语法混在一起使用：

```
.main-content {
    -webkit-box-ordinal-group: 2;   /*2009 版 - iOS 6-, Safari 3.1-6*/
    -moz-box-ordinal-group: 2;      /*2009 版 - Firefox 19-*/
    -ms-flex-order: 2;              /*2011 版 - IE 10*/
    -webkit-order: 2;               /*最新版 - Chrome*/
    order: 2;                       /*最新版 - Opera 12.1, Firefox 20+*/
    width: 60%;                     /*不会自动伸缩，其他列将占据空间*/
    -moz-box-flex: 1;               /*如果没有该声明，Firefox 19-将溢出 h，覆盖宽度*/
    background: white;
}
```

在新语法中，没有必要给边栏设置宽度，因为它们同样会使用 20%比例填充剩余的 40%空间。但是，如果不显式地设置宽度，在旧语法下会直接崩溃。

完成初步布局之后，需要重新编排列的顺序。这里设计主内容排列在中间，但在源码之中，它排在第一的位置。使用 Flexbox 可以非常容易实现，但是用户需要把 Flexbox 几种不同的语法混在一起使用：

```
.main-content {
    -webkit-box-ordinal-group: 2;
    -moz-box-ordinal-group: 2;
    -ms-flex-order: 2;
    -webkit-order: 2;
    order: 2;
}
.main-nav {
    -webkit-box-ordinal-group: 1;
    -moz-box-ordinal-group: 1;
    -ms-flex-order: 1;
    -webkit-order: 1;
```



```
        order: 1;
    }
    .main-sidebar {
        -webkit-box-ordinal-group: 3;
        -moz-box-ordinal-group: 3;
        -ms-flex-order: 3;
        -webkit-order: 3;
        order: 3;
    }
```



Note



视 频 讲 解

11.5.2　设计 3 行 3 列应用

本例借助 Flexbox 伸缩盒布局，设计页面呈现 3 行 3 列布局样式，同时能够根据窗口自适应调整各自空间，以满屏显示，效果如图 11.21 所示。

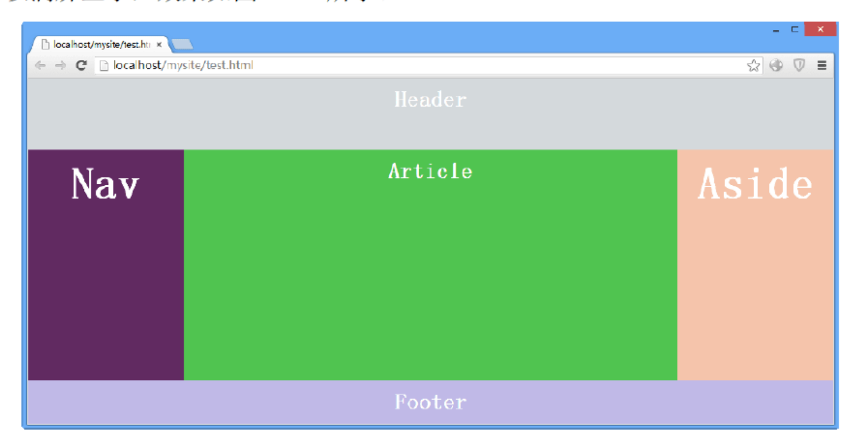


图 11.21　3 行 3 列布局样式

页面主要代码如下：

```
<style type="text/css">
/*基本样式*/
* {margin: 0; padding: 0;
    -moz-box-sizing: border-box;
    -webkit-box-sizing: border-box;
    box-sizing: border-box;
}
html, body {height: 100%; color: #fff;}
body {min-width: 100%;}
header, section, nav, aside, footer {display: block; text-align:center; font-size:2em; font-weight:bold;}
/*页眉框样式：限高、限宽*/
header {
    background-color: hsla(200,10%,70%,.5);
    min-height: 100px; padding: 10px 20px;
    min-width: 100%;
}
/*主体区域框样式：满宽显示*/
section {min-width: 100%;}
```





```
/*导航框样式：固定宽度*/
nav {background-color: hsla(300,60%,20%,.9);padding: 1%;width: 220px;}
/*文档栏样式*/
article {background-color: hsla(120,50%,50%,.9); padding: 1%;}
/*侧边栏样式：弹性宽度*/
aside {background-color: hsla(20,80%,80%,.9); padding: 1%;width: 220px;}
/*页脚样式：限高、限宽*/
footer {
    background-color: hsla(250,50%,80%,.9);
    min-height: 60px; padding: 1%;
    min-width: 100%;
}
/*flexbox 样式*/
body {
    /*设置 body 为伸缩容器*/
    display: -webkit-box;              /*旧版本：iOS 6-, Safari 3.1-6*/
    display: -moz-box;                 /*旧版本：Firefox 19-*/
    display: -ms-flexbox;              /*混合版本：IE10*/
    display: -webkit-flex;             /*新版本：Chrome*/
    display: flex;                     /*标准规范：Opera 12.1, Firefox 20+*/
    /*伸缩项目换行*/
    -moz-box-orient: vertical;
    -webkit-box-orient: vertical;
    -moz-box-direction: normal;
    -moz-box-direction: normal;
    -moz-box-lines: multiple;
    -webkit-box-lines: multiple;
    -webkit-flex-flow: column wrap;
    -ms-flex-flow: column wrap;
    flex-flow: column wrap;
}
/*实现 stick footer 效果*/
section {
    display: -moz-box;
    display: -webkit-box;
    display: -ms-flexbox;
    display: -webkit-flex;
    display: flex;
    -webkit-box-flex: 1;
    -moz-box-flex: 1;
    -ms-flex: 1;
    -webkit-flex: 1;
    flex: 1;
    -moz-box-orient: horizontal;
    -webkit-box-orient: horizontal;
    -moz-box-direction: normal;
    -webkit-box-direction: normal;
    -moz-box-lines: multiple;
    -webkit-box-lines: multiple;
    -ms-flex-flow: row wrap;
    -webkit-flex-flow: row wrap;
```



```
		flex-flow: row wrap;
		-moz-box-align: stretch;
		-webkit-box-align: stretch;
		-ms-flex-align: stretch;
		-webkit-align-items: stretch;
		align-items: stretch;
	}
	/*文章区域伸缩样式*/
	article {
		-moz-box-flex: 1;
		-webkit-box-flex: 1;
		-ms-flex: 1;
		-webkit-flex: 1;
		flex: 1;
		-moz-box-ordinal-group: 2;
		-webkit-box-ordinal-group: 2;
		-ms-flex-order: 2;
		-webkit-order: 2;
		order: 2;
	}
	/*侧边栏伸缩样式*/
	aside {
		-moz-box-ordinal-group: 3;
		-webkit-box-ordinal-group: 3;
		-ms-flex-order: 3;
		-webkit-order: 3;
		order: 3;
	}
	</style>

	<header>Header</header>
	<section>
		<article>Article</article>
		<nav>Nav</nav>
		<aside>Aside</aside>
	</section>
	<footer>Footer</footer>
```

11.6　在线练习

Flexbox 3 个不同版本的规范对应着不同的实现。需要关注哪个版本，取决于需要支持的浏览器。详细说明请扫码阅读。

在线练习



设计 CSS3 用户界面样式

2015 年 4 月，W3C 的 CSS 工作组发布 CSS 基本用户接口模块（CSS Basic User Interface Module Level 3，CSS3 UI）的标准工作草案，该文档描述了 CSS3 中对 HTML、XML 进行样式处理所需的与用户界面相关的 CSS 选择器、属性及属性值。CSS3 VI 模块负责控制与用户接口界面相关效果的呈现方式，它包含并扩展了在 CSS2 和 Selector 规范中定义的与用户接口有关的特性。

权威参考：http://www.w3.org/TR/css-ui-3/。

权威参考

【学习重点】

- 了解常用界面显示属性。
- 能够定义轮廓样式。
- 正确设计边框图像样式。
- 灵活设计圆角样式。
- 灵活设计阴影样式。



12.1　界 面 显 示



Note



视 频 讲 解

下面介绍 CSS3 用户界面的显示方式、显示大小和溢出处理问题。

12.1.1　显示方式

一般浏览器都支持两种显示模式：怪异模式和标准模式。在怪异模式下，border 和 padding 包含在 width 或 height 之内；在标准模式下，border、padding、width 或 height 是各自独立区域。

为了兼顾这两种解析模式，CSS3 定义了 box-sizing 属性，该属性能够定义对象尺寸的解析方式。box-sizing 属性的基本语法如下：

```
box-sizing: content-box | border-box;
```

取值简单说明如下。

☑　content-box：为默认值，padding 和 border 不被包含在定义的 width 和 height 之内。对象的实际宽度等于设置的 width 值和 border、padding 之和，即元素的宽度 = width + border + padding。

☑　border-box：padding 和 border 被包含在定义的 width 和 height 之内。对象的实际宽度就等于设置的 width 值，即使定义有 border 和 padding 也不会改变对象的实际宽度，即元素的宽度= width。

【示例】下面示例设计两个相同样式的盒子，在怪异模式和标准模式下的显示效果如图 12.1 所示。

```
<style type="text/css">
div {
    float: left;                    /*浮动显示*/
    height: 100px;                  /*元素的高度*/
    width: 100px;                   /*元素的宽度*/
    border: 50px solid red;         /*边框*/
    margin: 10px;                   /*外边距*/
    padding: 50px;                  /*内边距*/
}
.border-box {box-sizing: border-box;}   /*怪异模式解析*/
</style>
<div>标准模式</div>
<div class="border-box">怪异模式</div>
```

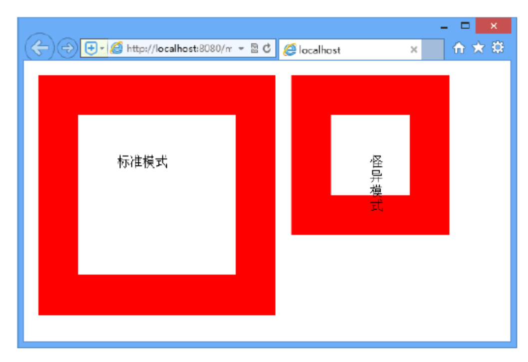


图 12.1　标准模式和怪异模式解析比较



从图 12.1 可以看到，在怪异模式下 width 属性值就是指元素的实际宽度，即 width 属性值中包含 padding 和 border 属性值。



Note



视频讲解

12.1.2 调整尺寸

为了增强用户体验，CSS3 增加了 resize 属性，允许用户通过拖动的方式改变元素的尺寸。resize 属性的基本语法如下：

```
resize:none | both | horizontal | vertical;
```

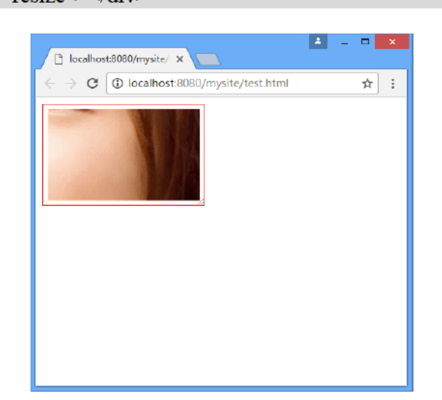
取值简单说明如下。

☑ none：为默认值，不允许用户调整元素大小。
☑ both：用户可以调节元素的宽度和高度。
☑ horizontal：用户可以调节元素的宽度。
☑ vertical：用户可以调节元素的高度。

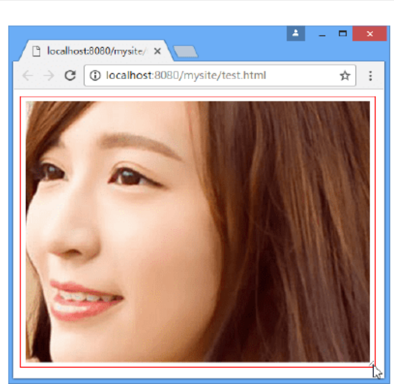
目前除了 IE 浏览器外，其他主流浏览器都基本支持该属性。

【示例】下面示例演示如何使用 resize 属性设计可以自由调整大小的图片，如图 12.2 所示。

```
<style type="text/css">
#resize {
     /*以背景方式显示图像，这样可以更轻松地控制缩放操作*/
    background:url(images/1.jpg) no-repeat center;
     /*设计背景图像仅在内容区域显示，留出补白区域*/
    background-clip:content;
     /*设计元素最小和最大显示尺寸，用户也只能够在该范围内自由调整*/
    width:200px; height:120px;
    max-width:800px; max-height:600px;
    padding:6px; border: 1px solid red;
     /*必须同时定义 overflow 和 resize，否则 resize 属性声明无效，元素默认溢出显示为 visible*/
    resize: both;
    overflow: auto;
}
</style>
<div id="resize"></div>
```



默认大小



鼠标拖动放大

图 12.2　调节元素尺寸







Note

12.1.3　缩放比例

zoom 是 IE 的专有属性，用于设置对象的缩放比例，另外它还可以触发 IE 的 haslayout 属性，具有清除浮动和 margin 重叠等作用，设计师常用这个属性解决 IE 浏览器存在的布局 Bug。

CSS3 支持该属性，基本语法如下：

```
zoom: normal | <number> | <percentage>
```

取值说明如下。

☑　normal：使用对象的实际尺寸。

☑　<number>：用浮点数来定义缩放比例，不允许为负值。

☑　<percentage>：用百分比来定义缩放比例，不允许为负值。

目前，除了 Firefox 浏览器之外，所有主流浏览器都支持该属性。

【示例】 下面示例使用 zoom 放大第 2 幅图片为原来的 2 倍，比较效果如图 12.3 所示。

```
<style type="text/css">
img {
    height: 200px;
    margin-right: 6px;
}
img.zoom {zoom: 2;}
</style>
<img src="images/bg.jpg"/>
<img class="zoom" src="images/bg.jpg"/>
```



图 12.3　放大图片显示尺寸

当 zoom 属性值为 1.0 或 100%时，相当于 normal，表示不缩放。小于 1 的正数，表示缩小，如"zoom: 0.5;"表示缩小为原来的二分之一。

12.2　轮 廓 样 式

轮廓与边框不同，它不占用空间，且不一定是矩形。轮廓属于动态样式，只有当对象获取焦点或者被激活时呈现，如按钮、活动窗体域、图形地图等周围添加一圈轮廓线，使对象突出显示。



视频讲解



Note

12.2.1 定义轮廓

outline 属性可以定义块元素的轮廓线，该属性在 CSS2.1 规范中已被明确定义，但是并未得到各主流浏览器的广泛支持，CSS3 增强了该特性。outline 属性的基本语法如下：

```
outline: <'outline-width'> || <'outline-style'> || <'outline-color'> || <'outline-offset'>
```

取值简单说明如下。

☑ <'outline-width'>：指定轮廓边框的宽度。

☑ <'outline-style'>：指定轮廓边框的样式。

☑ <'outline-color'>：指定轮廓边框的颜色。

☑ <'outline-offset'>：指定轮廓边框的偏移值。

注意： outline 创建的轮廓线是画在一个框“上面”，也就是说，轮廓线总是在顶上，不会影响该框或任何其他框的尺寸。因此，显示或不显示轮廓线不会影响文档流，也不会破坏网页布局。轮廓线可能是非矩形的。例如，如果元素被分割在好几行，那么轮廓线就至少是能要包含该元素所有框的外廓。和边框不同的是，外廓在线框的起止都不是开放的，它总是完全闭合的。

【示例】 下面示例设计当文本框获得焦点时，在周围画一个粗实线外廓，提醒用户交互效果，效果如图 12.4 所示。

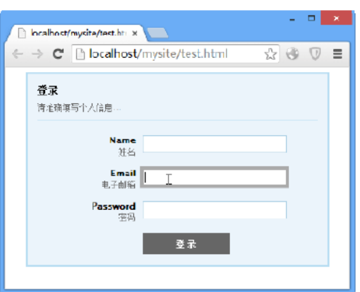
```
<style type="text/css">
/*统一页面字体和大小*/
body {
    font-family:"Lucida Grande", "Lucida Sans Unicode", Verdana, Arial, Helvetica, sans-serif;
    font-size:12px;
}
/*清除常用元素的边界、补白、边框默认样式*/
p, h1, form, button {border:0; margin:0; padding:0;}
/*定义一个强制换行显示类*/
.spacer {clear:both; height:1px;}
/*定义表单外框样式*/
.myform {margin:0 auto; width:400px; padding:14px;}
/*定制当前表单样式*/
#stylized {border:solid 2px #b7ddf2; background:#ebf4fb;}
/*设计表单内 div 和 p 通用样式效果*/
#stylized h1 {font-size:14px; font-weight:bold;margin-bottom:8px;}
#stylized p {
    font-size:11px; color:#666666;
    margin-bottom:20px; padding-bottom:10px;
    border-bottom:solid 1px #b7ddf2;
}
#stylized label {/*定义表单标签样式*/
    display:block; width:140px;
    font-weight:bold; text-align:right;
    float:left;
}
/*定义小字体样式类*/
```



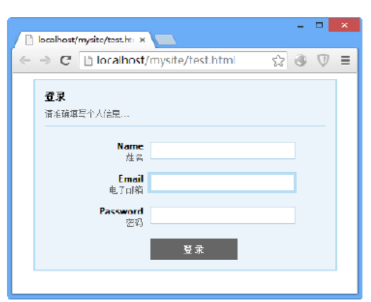
```
    #stylized .small {
        color:#666666; font-size:11px; font-weight:normal; text-align:right;
        display:block; width:140px;
    }
    /*统一输入文本框样式*/
    #stylized input {
        float:left;
        font-size:12px;
        padding:4px 2px; margin:2px 0 20px 10px;
        border:solid 1px #aacfe4; width:200px;
    }
    /*定义图形化按钮样式*/
    #stylized button {
        clear:both;
        margin-left:150px;
        width:125px; height:31px;
        background:#666666 url(images/button.png) no-repeat;
        text-align:center; line-height:31px; color:#FFFFFF; font-size:11px; font-weight:bold;
    }
    /*设计表单内文本框和按钮在被激活和获取焦点状态下时，轮廓线的宽、样式和颜色*/
    input:focus, button:focus {outline: thick solid #b7ddf2}
    input:active, button:active    {outline: thick solid #aaa}
    </style>
    <div id="stylized" class="myform">
        <form id="form1" name="form1" method="post" action="">
            <h1>登录</h1>
            <p>请准确填写个人信息...</p>
            <label>Name <span class="small">姓名</span> </label>
            <input type="text" name="textfield" id="textfield" />
            <label>Email <span class="small">电子邮箱</span> </label>
            <input type="text" name="textfield" id="textfield" />
            <label>Password <span class="small">密码</span> </label>
            <input type="text" name="textfield" id="textfield" />
            <button   type="submit">登  录</button>
            <div class="spacer"></div>
        </form>
    </div>
```



默认状态

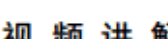


激活状态



获取焦点状态

图 12.4　设计文本框的轮廓线


Note

12.2.2 设计轮廓线

CSS3 为轮廓定义了很多属性，使用这些属性可以设计轮廓线样式。

1．设置宽度

outline-width 属性可以设置轮廓线的宽度。基本语法如下：

```
outline-width: <length> | thin | medium | thick
```

取值简单说明如下。

☑ <length>：定义轮廓粗细的值。
☑ thin：定义细轮廓。
☑ medium：定义中等的轮廓，为默认值。
☑ thick：定义粗的轮廓。

注意： 只有当轮廓样式不是 none 时，该属性才会起作用。如果样式为 none，宽度实际上会重置为 0。不允许设置为负值。

2．设置样式

outline-style 属性可以设置轮廓线的样式。基本语法如下：

```
outline-style: none | dotted | dashed | solid | double | groove | ridge | inset | outset
```

取值简单说明如下。

☑ none：无轮廓，为默认值。
☑ dotted：点状轮廓。
☑ dashed：虚线轮廓。
☑ solid：实线轮廓。
☑ double：双线轮廓。两条单线与其间隔的和等于指定 outline-width 值。
☑ groove：3D 凹槽轮廓。
☑ ridge：3D 凸槽轮廓。
☑ inset：3D 凹边轮廓。
☑ outset：3D 凸边轮廓。

3．设置颜色

outline-color 属性可以设置轮廓线的颜色。基本语法如下：

```
outline-color: <color> | invert
```

取值简单说明如下。

☑ <color>：指定颜色。
☑ invert：使用背景色的反色。该参数值目前仅在 IE 及 Opera 下有效。

4．设置偏移

outline-offset 属性可以设置轮廓线的偏移位置。基本语法如下：

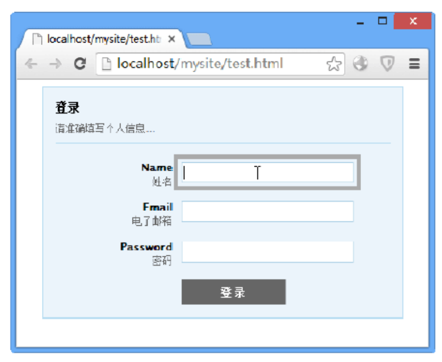
```
outline-offset: <length>
```



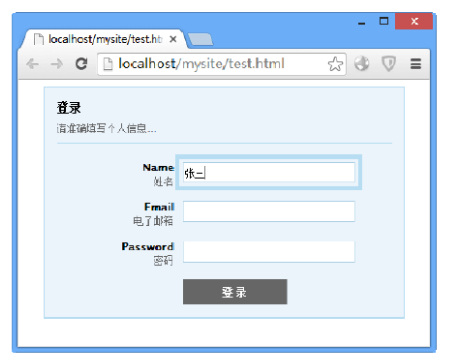
用长度值来定义轮廓偏移，允许为负值，默认值为 0。

【示例 1】在 12.2.1 节示例基础上，通过 outline-offset 属性放大轮廓线，使其看起来更大方，演示效果如图 12.5 所示。下面代码仅显示局部 CSS 样式，完整示例样式和结构请参考 12.2.1 节示例代码。

```
<style type="text/css">
…
/*设计表单内文本框和按钮在被激活和获取焦点状态下时，轮廓线的宽、样式和颜色*/
input:focus, button:focus {outline: thick solid #b7ddf2}
input:active, button:active   {outline: thick solid #aaa}
/*通过 outline-offset 属性放大轮廓线*/
input:active, button:active {outline-offset: 4px;}
input:focus, button:focus {outline-offset: 4px;}
</style>
<div id="stylized" class="myform">
    <form id="form1" name="form1" method="post" action="">
        <h1>登录</h1>
        …
    </form>
</div>
```



激活状态



获取焦点状态

图 12.5　放大激活和焦点提示框

【示例 2】下面示例为段落文本中部分文字定义轮廓线，演示效果如图 12.6 所示。

```
<style type="text/css">
.outline {outline: red solid 2px;}
</style>
<p><b>注释：</b>只有在规定了 !DOCTYPE 时，<span class="outline">Internet Explorer 8 （以及更高版本）
</span>才支持 outline 属性。</p>
```

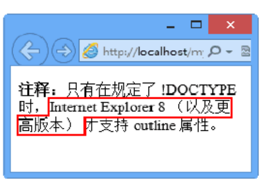


图 12.6　轮廓边框效果





12.3 边 框 样 式

边框是 CSS 盒模型的重要组成部分，本节将介绍 CSS3 增强的边框样式，包括图像边框和圆角边框。

权威参考：http://www.w3.org/TR/css3-border/。


权威参考


视频讲解

12.3.1 定义边框图像源

CSS3 新增的 border-image 属性能够模拟 background-image 属性功能，且功能更加强大，该属性的基本语法如下：

```
border-image: <'border-image-source'> || <'border-image-slice'> [/ <'border-image-width'> | / <'border- image-width'>? / <'border-image-outset'>]? || <'border-image-repeat'>
```

取值说明如下。

☑ <'border-image-source'>：设置对象的边框是否用图像定义样式或图像来源路径。

☑ <'border-image-slice'>：设置边框图像的分割方式。

☑ <'border-image-width'>：设置对象的边框图像的宽度。

☑ <'border-image-outset'>：设置对象的边框图像的扩展。

☑ <'border-image-repeat'>：设置对象的边框图像的平铺方式。

【示例】下面示例为元素 div 定义边框图像，使用 border-image-source 导入外部图像源 images/border1.png，通过 border-image-slice 属性，值为(27 27 27 27)，把图像切分为 9 块，然后分别把这 9 块图像切片按顺序填充到边框四边、四角和内容区域。示例主要代码如下，页面浏览效果如图 12.7 所示。

```
<style type="text/css">
div {
    height:160px;
    border:solid 27px;
    /*设置边框图像*/
    border-image: url(images/border1.png) 27;
</style>
<div></div>
```

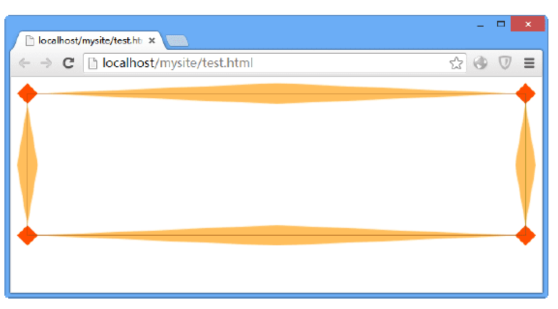


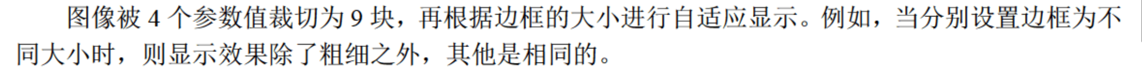
图 12.7　定义边框背景样式

在上面示例中，使用了一个 71px*71px 大小的图像，将这个正方形图像等分为 9 个方块，每个方



块大小为 21px*21px。当声明 border-image-slice 属性值为(27 27 27 27)时，则按下面说明进行解析：

☑　第 1 个参数值表示从上向下裁切图像，显示在顶边。
☑　第 2 个参数值表示从右向左裁切图像，显示在右边。
☑　第 3 个参数值表示从下向上裁切图像，显示在底边。
☑　第 4 个参数值表示从左向右裁切图像，显示在左边。



图像被 4 个参数值裁切为 9 块，再根据边框的大小进行自适应显示。例如，当分别设置边框为不同大小时，则显示效果除了粗细之外，其他是相同的。



Note



视 频 讲 解

12.3.2　定义边框图像平铺方式

border-image-repeat 属性设置对象的边框图像的平铺方式。该属性的基本语法如下：

```
border-image-repeat: [stretch | repeat | round | space]{1,2}
```

取值简单说明如下。

☑　stretch：用拉伸方式来填充边框图像，为默认值。
☑　repeat：用平铺方式来填充边框图像。当图片碰到边界时，如果超过则被截断。
☑　round：用平铺方式来填充边框图像。图像会根据边框的尺寸动态调整图像的大小，直至正好可以铺满整个边框。
☑　space：用平铺方式来填充边框图像。图像会根据边框的尺寸动态调整图像之间的间距，直至正好可以铺满整个边框。

【示例】下面示例以 12.3.1 节示例为基础，设置边框图像平铺显示：border-image-repeat:round;，演示效果如图 12.8 所示。

```
<style type="text/css">
div {
    height:160px;
    background:hsla(93,96%,62%,1.00);
    border:solid 27px red;
    /*设置边框图像源*/
    border-image-source: url(images/border1.png);
    /*设置边框图像的平铺方式*/
    border-image-repeat:round;
}
</style>
```

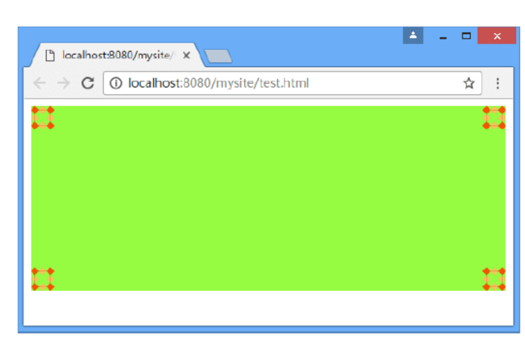
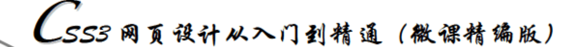
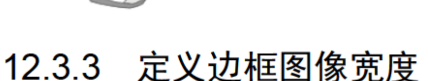


图 12.8　定义边框图像平铺显示




视频讲解


Note



12.3.3　定义边框图像宽度

border-image-width 属性设置对象的边框图像的宽度。该属性的基本语法如下：

```
border-image-width: [<length> | <percentage> | <number> | auto]{1,4}
```

取值简单说明如下。

☑ <length>：用长度值指定宽度。不允许为负值。

☑ <percentage>：用百分比指定宽度。参照其包含块进行计算，不允许为负值。

☑ <number>：用浮点数指定宽度。不允许为负值。

☑ auto：如果 auto 值被设置，则<'border-image-width'>采用与<'border-image-slice'>相同的值。

【示例】下面示例以 12.3.2 节示例为基础，设置边框背景平铺显示：border-image-repeat:round;，图像宽度为 500px，演示效果如图 12.9 所示。

```
<style type="text/css">
div {
    height:160px;
    background:hsla(93,96%,62%,1.00);
    border:solid 27px red;
    /*设置边框图像源*/
    border-image-source: url(images/border1.png);
    /*设置边框图像的平铺方式*/
    border-image-repeat:round;
    /*设置边框图像的宽度*/
    border-image-width: 500px;
}
</style>
<div>border-image-source: url(images/border1.png);<br>
    border-image-repeat:round;<br>
    border-image-width:500px;</div>
```



图 12.9　定义边框图像宽度


视频讲解

12.3.4　定义边框图像分割方式

border-image-slice 属性设置对象的边框图像的分割方式。该属性的基本语法如下：

```
border-image-slice: [<number> | <percentage>]{1,4} && fill?
```



取值简单说明如下。

☑ <number>：用浮点数指定宽度。不允许为负值。

☑ <percentage>：用百分比指定宽度。参照其包含块区域进行计算，不允许为负值。

☑ fill：保留裁剪后的中间区域，其铺排方式遵循<' border-image-repeat '>的设定。



【示例】下面示例以 12.3.3 节示例为基础，设置边框背景平铺显示：border-image-repeat:round;，设置裁切值为 10：border-image-slice: 10;，演示效果如图 12.10 所示。

```
<style type="text/css">
div {
    height:160px;
    background:hsla(93,96%,62%,1.00);
    border:solid 27px red;
    /*设置边框图像源*/
    border-image-source: url(images/border1.png);
    /*设置边框图像的平铺方式*/
    border-image-repeat:round;
    /*设置边框图像的宽度*/
    border-image-width: 500px;
    /*设置边框图像的裁切值为 10*/
    border-image-slice: 10;
}
</style>
<div>border-image-source: url(images/border1.png);<br>
    border-image-repeat:round;<br>
    border-image-slice: 10;</div>
```

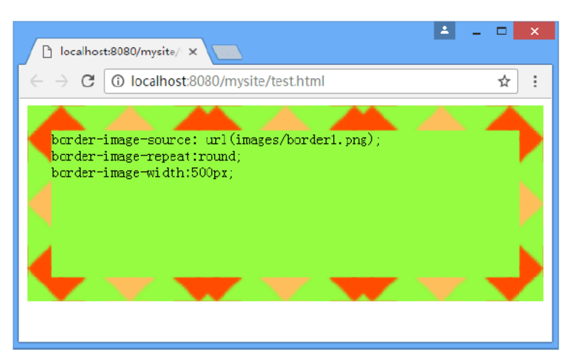
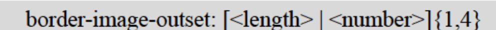


图 12.10 定义边框图像裁切值

12.3.5 定义边框图像扩展



border-image-outset 属性设置对象的边框图像的扩展。该属性的基本语法如下：



```
border-image-outset: [<length> | <number>]{1,4}
```

取值简单说明如下。

☑ <length>：用长度值指定宽度。不允许为负值。

☑ <number>：用浮点数指定宽度。不允许为负值。

【示例】下面以 12.3.4 节示例为基础，设置边框背景向外扩展 50px，演示效果如图 12.11 所示。



Note

```
<style type="text/css">
div {
    height:160px;
    margin:60px;
    background:hsla(93,96%,62%,1.00);
    border:solid 27px red;
    /*设置边框图像源*/
    border-image-source: url(images/border1.png);
    /*设置边框图像的平铺方式*/
    border-image-repeat:round;
    /*设置边框图像的宽度*/
    border-image-width: 500px;
    /*设置边框图像的裁切值为 10*/
    border-image-slice: 10;
    /*设置边框图像向外扩展 50px*/
    border-image-outset: 50px;
}
</style>
<div>border-image-source: url(images/border1.png);<br>
    border-image-repeat:round;<br>
    border-image-slice: 10;<br>
    border-image-outset: 50px;</div>
```

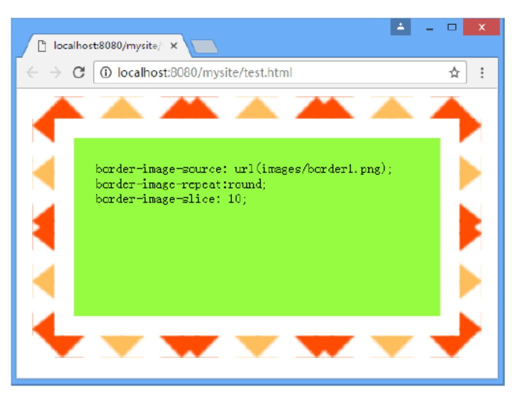


图 12.11　定义边框图像向外扩展



视频讲解

12.3.6　案例：应用边框图像

下面结合示例介绍 border-image 在页面中的应用。

【示例 1】下面示例演示如何设计左下和右下圆角显示，演示效果如图 12.12 所示。

```
<style type="text/css">
div {
    height: 120px;
    text-align:center;
    border-style:solid;
    border-width: 10px;
    border-image: url(images/r2.png) 20;
}
```



```
</style>
<div>border-image: url(images/r2.png) 20; 图像源效果图下所示：<br>
    <img src="images/r2.png" /></div>
```



【示例 2】设计完全圆角边框效果。设计圆角图像大小为 42px*42px，裁切半径为 20px，显示效果如图 12.13 所示。

```
<style type="text/css">
div {
    height: 120px;
    text-align:center;
    border-style:solid;
    border-width: 10px;
    border-image: url(images/r3.png) 20;
}
</style>
<div>border-image: url(images/r3.png) 20; 图像源效果图下所示：<br>
    <img src="images/r3.png" /></div>
```



图 12.12　定义边框局部圆角样式



图 12.13　定义完全圆角边框样式

【示例 3】设计阴影特效。设计边框图像大小为 42px*42px，显示效果如图 12.14 所示。

```
<style type="text/css">
img {
    height:400px;
    border-style:solid;
    border-width:2px 5px 6px 2px;
    border-image: url(images/r4.png)   2 5 6 2;
}
</style>

<img src="images/2.jpg" />
```

【示例 4】设计选项卡。设计边框图像大小为 12px*27px，圆角半径为 12px，显示效果如图 12.15 所示。

```
<style type="text/css">
ul{
    margin:0; padding:0;
    list-style-type:none;
}
li {
    width:100px; height:20px;
    border-style:solid;
```



```
        cursor:pointer;
        float:left;
        padding:4px 0;
        text-align:center;
        border-width:5px 5px 0px;
        border-image: url(images/r5.png) 5 5 0;
}
</style>
<ul>
    <li>首页</li>
    <li>咨询</li>
    <li>关于</li>
</ul>
```

Note



图 12.14　定义边框阴影样式

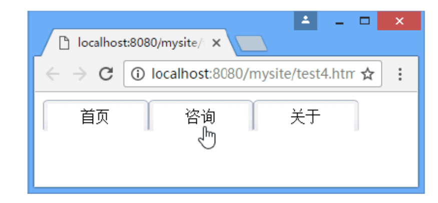


图 12.15　定义选项卡样式



视频讲解

12.3.7　定义圆角边框

CSS3 新增 border-radius 属性，使用它可以设计元素的边框以圆角样式显示。border-radius 属性的基本语法如下：

```
border-radius: [<length> | <percentage>]{1,4} [/ [<length> | <percentage>]{1,4}]?
```

取值简单说明如下。

☑ <length>：用长度值设置对象的圆角半径长度。不允许为负值。

☑ <percentage>：用百分比设置对象的圆角半径长度。不允许为负值。

为了方便定义 4 个顶角的圆角，border-radius 属性派生了 4 个子属性。

☑ border-top-right-radius：定义右上角的圆角。

☑ border-bottom-right-radius：定义右下角的圆角。

☑ border-bottom-left-radius：定义左下角的圆角。

☑ border-top-left-radius：定义左上角的圆角。

提示： border-radius 属性可包含两个参数值：第一个值表示圆角的水平半径，第二个值表示圆角的垂直半径，两个参数值通过斜线分隔。如果仅包含一个参数值，则第二个值与第一个值相同，它表示这个角是一个四分之一圆角。如果参数值中包含 0，则这个角就是矩形，不会显示为圆角。





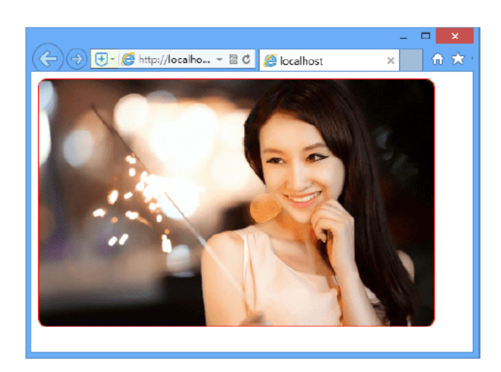
针对 border-radius 属性参数值，各种浏览器的处理方式并不一致。在 Chrome 和 Safari 浏览器中，会绘制出一个椭圆形边框，第一个半径为椭圆的水平方向半径，第二个半径为椭圆的垂直方向半径。在 Firefox 和 Opera 浏览器中，将第一个半径作为边框左上角与右下角的圆半径来绘制，将第二个半径作为边框右上角与左下角的圆半径来绘制。

【示例 1】下面给 border-radius 属性设置一个值：10px，则演示效果如图 12.16 所示。

```
<style type="text/css">
img {
    height:300px;
    border:1px solid red;
    border-radius:10px;
}
</style>
<img src="images/1.jpg" />
```

如果为 border-radius 属性设置两个参数值，则效果如图 12.17 所示。

```
img {
    height:300px;
    border:1px solid red;
    border-radius:20px/40px;
}
```



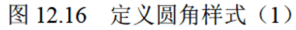


图 12.16　定义圆角样式（1）

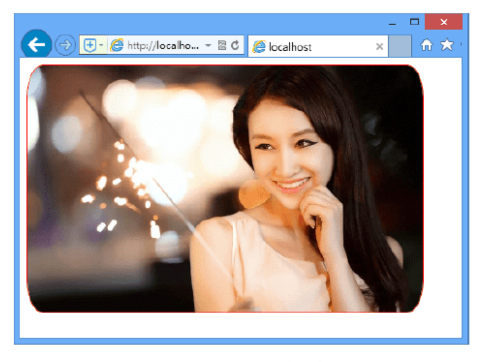


图 12.17　定义圆角样式（2）

也可以为元素的 4 个顶角定义不同的值，实现的方法有两种。

第 1 种，利用 border-radius 属性，为其赋一组值。当为 border-radius 属性赋一组值时，将遵循 CSS 赋值规则，可以包含 2 个、3 个或者 4 个值集合。但是此时无法使用斜杠方式定义圆角的水平和垂直半径。

如果是 4 个值，则这 4 个值将按照 top-left、top-right、bottom-right、bottom-left 顺序来设置。

如果 bottom-left 值省略，那么它等于 top-right。

如果 bottom-right 值省略，那么它等于 top-left。

如果 top-right 值省略，那么它等于 top-left。

如果为 border-radius 属性设置 4 个值的集合参数，则每个值表示每个角的圆角半径。

【示例 2】下面示例为图像的 4 个顶角定义不同圆角半径，演示效果如图 12.18 所示。

```
img {
    height:300px;
```



```
        border:1px solid red;
        border-radius:10px 30px 50px 70px;
    }
```



Note

如果为 border-radius 属性设置 3 个值的集合参数，则第 1 个值表示左上角的圆角半径，第 2 个值表示右上和左下两个角的圆角半径，第 3 个值表示右下角的圆角半径。

如果为 border-radius 属性设置 2 个值的集合参数，则第 1 个值表示左上角和右下角的圆角半径，第 2 个值表示右上和左下两个角的圆角半径。

第 2 种，利用派生子属性进行定义，如 border-top-right-radius、border-bottom-right -radius、border-bottom-left-radius、border-top-left-radius。

注意：Gecko 和 Presto 引擎在写法上存在很大差异。

【示例 3】下面代码定义 div 元素右上角为 50px 的圆角，演示效果如图 12.19 所示。

```
    img {
        height:300px;
        border:1px solid red;
        -moz-border-radius-topright:50px;
        -webkit-border-top-right-radius:50px;
        border-top-right-radius:50px;
    }
```

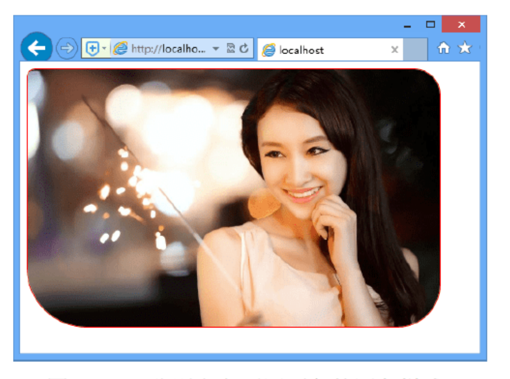


图 12.18　分别定义不同顶角的圆角样式

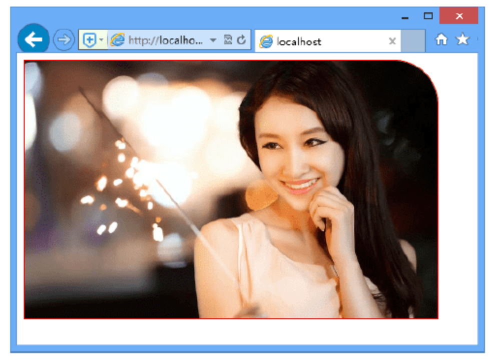


图 12.19　定义某个顶角的圆角样式



视频讲解

12.3.8　案例：设计椭圆图形

使用 border-radius 属性设计圆角时，可能会存在下面几种情况：

☑　如果受影响的角的两个相邻边宽度不同，那么这个圆角将会从宽的一边圆滑过渡到窄的一边，即偏向宽边的圆弧略大，而偏向窄边的圆弧略小。

☑　如果两条边宽度相同，那么圆角两个相邻边呈对称圆弧显示，即相交 45° 的对称线上。

☑　如果一条边宽度是相邻另一条边宽度的两倍，那么两边圆弧线交于靠近窄边的 30° 角线上。

border-radius 不允许圆角彼此重叠，当相邻两个圆角的半径之和大于元素的宽或高时，在浏览器中会呈现为椭圆或正圆效果。

【示例】下面代码定义 img 元素显示为圆形，当图像宽高比不同时，显示效果不同，如图 12.20 所示。






```
<style type="text/css">
img {/*定义图像圆角边框*/
    border: solid 1px red;
    border-radius: 50%; /*圆角*/
}
.r1 {/*定义第 1 幅图像宽高比为 1:1*/
    width:300px;
    height:300px;
}
.r2 {/*定义第 2 幅图像宽高比不为 1:1*/
    width:300px;
    height:200px;
}
.r3 {/*定义第 3 幅图像宽高比不为 1:1*/
    width:300px;
    height:100px;
    border-radius: 20px; /*定义圆角*/
}
</style>
<img class="r1" src="images/1.jpg" title="圆角图像" />
<img class="r2" src="images/1.jpg" title="椭圆图像" />
<img class="r3" src="images/1.jpg" title="圆形图像" />
```

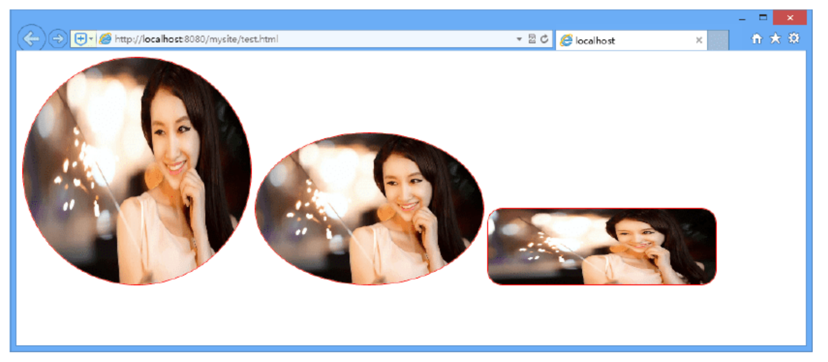


图 12.20　定义圆形显示的效果

12.4　盒 子 阴 影

CSS3 的 box-shadow 类似于 text-shadow，不过 text-shadow 负责为文本设置阴影，而 box-shadow 负责给对象定义图层阴影效果。

权威参考：http://www.w3.org/TR/css3-border/。

权威参考　视频讲解

12.4.1　定义盒子阴影

box-shadow 属性可以定义元素的阴影，基本语法如下：

```
box-shadow: none | inset? && <length>{2,4} && <color>?
```



Note

取值简单说明如下。

☑ none：无阴影。

☑ inset：设置对象的阴影类型为内阴影。该值为空时，则对象的阴影类型为外阴影。

☑ <length>①：第 1 个长度值用来设置对象的阴影水平偏移值。可以为负值。

☑ <length>②：第 2 个长度值用来设置对象的阴影垂直偏移值。可以为负值。

☑ <length>③：如果提供了第 3 个长度值，则用来设置对象的阴影模糊值。不允许为负值。

☑ <length>④：如果提供了第 4 个长度值，则用来设置对象的阴影外延值。可以为负值。

☑ <color>：设置对象的阴影的颜色。

下面结合示例进行演示说明。

【示例 1】下面示例定义一个简单的实影投影效果，演示效果如图 12.21 所示。

```
<style type="text/css">
img{
    height:300px;
    box-shadow:5px 5px;
}
</style>
<img src="images/1.jpg" />
```

【示例 2】定义位移、阴影大小和阴影颜色，演示效果如图 12.22 所示。

```
img{
    height:300px;
    box-shadow:2px 2px 10px #06C;
}
```



图 12.21　定义简单的阴影效果



图 12.22　定义复杂的阴影效果

【示例 3】定义内阴影，阴影大小为 10px，颜色为#06C，演示效果如图 12.23 所示。

```
<style type="text/css">
pre {
    padding: 26px;
    font-size:24px;
    box-shadow: inset 2px 2px 10px #06C;
}
</style>
<pre>
```



```
-moz-box-shadow: inset 2px 2px 10px #06C;
-webkit-box-shadow: inset 2px 2px 10px #06C;
box-shadow: inset 2px 2px 10px #06C;
</pre>
```

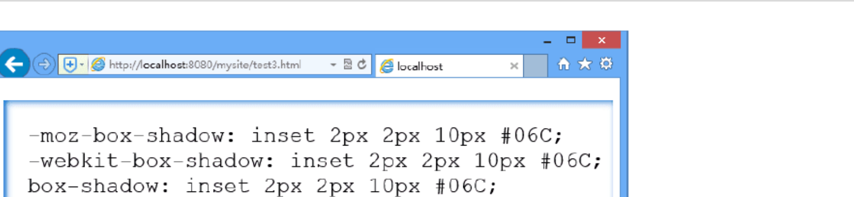


图 12.23　定义内阴影效果

【示例 4】通过设置多组参数值定义多色阴影，演示效果如图 12.24 所示。

```
img {
    height: 300px;
    box-shadow: -10px 0 12px red,
                10px 0 12px blue,
                0 -10px 12px yellow,
                0 10px 12px green;
}
```

【示例 5】通过多组参数值定义渐变阴影，演示效果如图 12.25 所示。

```
<!doctype html>
img{
    height:300px;
    box-shadow:0 0 10px red,
               2px 2px 10px 10px yellow,
               4px 4px 12px 12px green;
}
```



图 12.24　定义多色阴影效果

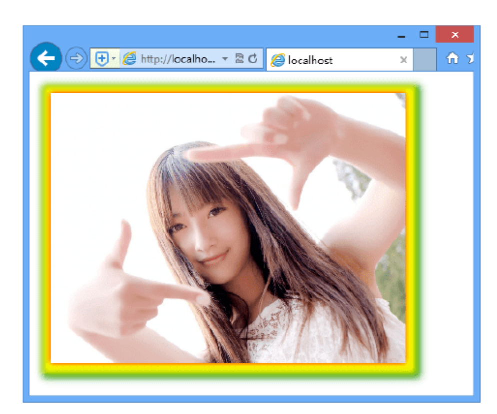


图 12.25　定义渐变阴影效果

注意：当给同一个元素设计多个阴影时，最先写的阴影将显示在最顶层。


视频讲解


Note

12.4.2 案例：box-shadow 的应用

本节通过一个简单的示例进一步练习 box-shadow 的应用。

【操作步骤】

第 1 步，设计一个简单的盒子，并定义基本形状。

```
<style type="text/css">
.box{
    width:100px; height:100px;              /*固定大小*/
    text-align:center; line-height:100px;   /*显示在中央*/
    background-color:rgba(255,204,0,.5);    /*浅色背景*/
    border-radius:10px;                     /*适当圆角*/
    padding:10px; margin:10px;              /*添加间距*/
}
</style>
<div class="box bs1">box-shadow</div>
```

第 2 步，阴影就是对原对象的复制，包括内边距和边框都属于 box 的占位范围，阴影也包括对内边距和边框的复制，但是阴影本身不占据布局的空间。比较效果如图 12.26 所示。

```
.bs1{box-shadow:120px 0px #ccc;}
```

第 3 步，设计四周有一样模糊值的阴影，如图 12.27 所示。

```
.bs1{box-shadow:0 0 20px #666;}
```

第 4 步，定义 5px 扩展阴影，如图 12.28 所示。

```
.bs1{box-shadow:0 0 0 5px #333;}
```



图 12.26　比较对象和阴影大小

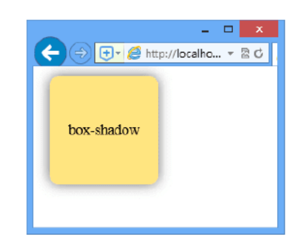


图 12.27　四周同时显示阴影

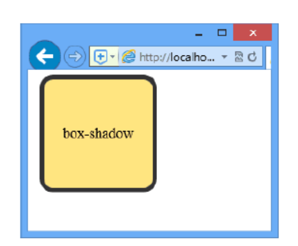


图 12.28　定义扩展阴影

阴影不像 border 要占据布局的空间，因此要实现对象鼠标经过产生外围的边框，可以使用阴影的扩展来代替 border。或者使用 border 的 transparent 实现，不过不如 box-shadow 的 spread 扩展方便。如果使用 border，布局会产生影响。

第 5 步，扩展为负值的阴影，如图 12.29 所示。

```
.bs1 {
    box-shadow: 0 15px 10px -15px #333;
    border: none;
}
```

注意： 要产生这样的效果，y 轴的值和 spread 的值刚好是一样且相反的。其他边设计同理。





第 6 步，定义内阴影，如图 12.30 所示。

```
.bs1 {
    background-color: #1C8426;
    box-shadow: 0px 0px 20px #fff inset;
}
```

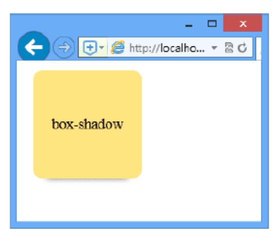


图 12.29 定义负值阴影

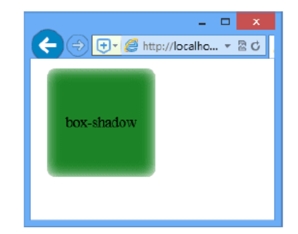


图 12.30 定义内阴影

注意： 可以直接为 div 这样的盒子设置 box-shadow 盒阴影，但是不能直接为 img 图片设置盒阴影。

```
/*直接在图片上添加内阴影，无效*/
.img-shadow img {
    box-shadow: inset 0 0 20px red;
}
```

可以通过为 img 的容器 div 设置内阴影，然后让 img 的 z-index 为-1，来解决这个问题。但是这种做法不可以为容器设置背景颜色，因为容器的级别比图片高，设置了背景颜色会挡住图片，效果如图 12.31 所示。

```
/*在图片容器上添加内阴影，生效*/
.box-shadow {
    box-shadow: inset 0 0 20px red;
    display:inline-block;
}
.box-shadow img {
    position: relative;
    z-index: -1;
}
```

还有一个更好的方法，不用考虑图片的层级，利用:before 伪元素可以实现，而且还可以为父容器添加背景颜色等。

```
/*为图片容器添加伪元素或伪类，不用为 img 设置负的 z-index 值，有内阴影*/
img {
    position: relative;
    background-color: #FC3;
    padding: 5px;
}
img:before {
    content: '';
    position: absolute;
    top: 0; right: 0; bottom: 0;    left: 0;
    box-shadow: inset 0 0 40px #f00;
}
```



Note

第 7 步，定义多个阴影，如图 12.32 所示。

```
.bs1 {
    box-shadow: 40px 40px rgba(26,202,221,0.5),
    80px 80px rgba(236,43,120,.5);
    border-radius: 0;
}
```

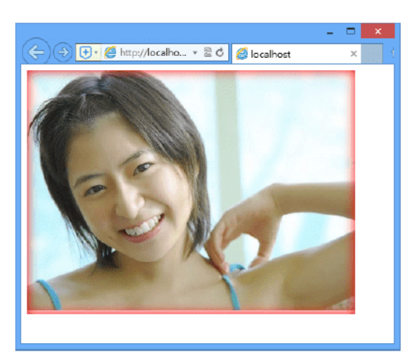


图 12.31　为图片定义内阴影

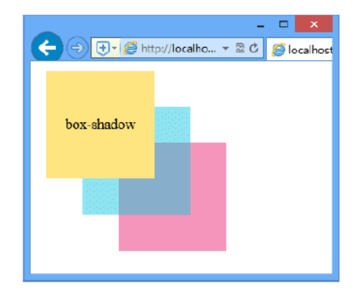


图 12.32　定义多个阴影

提示：阴影也是有层叠关系的，前面的阴影层级高，会压住后面的阴影。阴影和阴影之间的透明度可见，而主体对象的透明度对阴影不起作用。



视频讲解

12.4.3　案例：设计翘边阴影

本例使用 box-shadow 设计翘边阴影，翘边效果就是四角旁边翘起阴影，如图 12.33 所示。

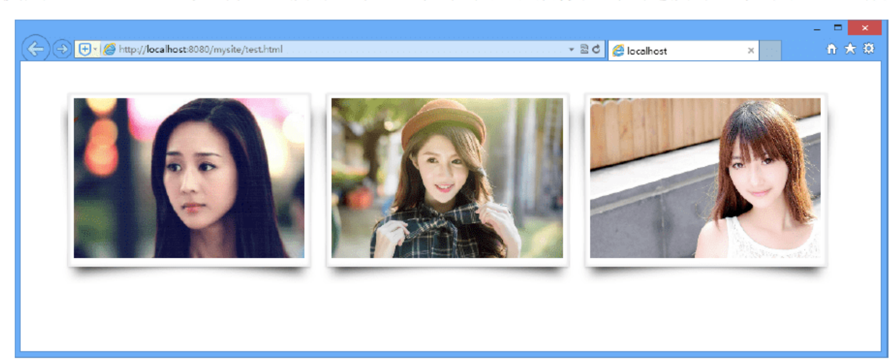


图 12.33　设计翘边阴影效果

示例主要代码如下：

```
<style type="text/css">
* {margin: 0; padding: 0;}              /*清除页边距*/
ul {list-style: none;}                  /*清除项目列表符号*/
.box {/*设计盒子样式*/
    width: 980px; height: auto;         /*固定大小，高度自动调整*/
    clear: both;
    overflow: hidden;                   /*禁止超出显示*/
```



```
        margin: 20px auto;                  /*居中显示*/
    }
    .box li {/*设计每个图片外框样式*/
        background: #fff;                   /*白色背景*/
        float: left;                        /*浮动并列显示*/
        position: relative;                 /*定义定位包含框*/
        margin: 20px 10px;                  /*调整项目间距*/
        border: 2px solid #efefef;          /*增加浅色边框*/
        /*添加内阴影*/
        box-shadow: 0 1px 4px rgba(0,0,0,0.27), 0 0 4px rgba(0,0,0,0.1) inset;
    }
    .box li img {/*固定图片大小，增加外边距*/
        width: 290px; height: 200px;
        margin: 5px;
    }
    .box li:before {/*在左侧添加翘起阴影*/
        content: "";                        /*空内容*/
        position: absolute;                 /*固定定位*/
        width: 90%; height: 80%;            /*定义大小*/
        bottom: 13px; left: 21px;           /*定位*/
        background: transparent;            /*透明背景*/
        z-index: -2;                        /*显示在照片下面*/
        box-shadow: 0 8px 20px rgba(0,0,0,0.8); /*添加阴影*/
        transform: skew(-12deg) rotate(-6deg);  /*变形并旋转阴影，让其翘起*/
    }
    .box li:after {/*在右侧添加翘起阴影，方法同上*/
        content: "";
        position: absolute;
        width: 90%;height: 80%;
        bottom: 13px; right: 21px;
        z-index: -2;
        background: transparent; box-shadow: 0 8px 20px rgba(0,0,0,0.8);
        transform: skew(12deg) rotate(6deg);
    }
    </style>
    <ul class="box">
        <li><img src="images/1.jpg" /></li>
        <li><img src="images/2.jpg" /></li>
        <li><img src="images/3.jpg" /></li>
    </ul>
```



本例主要使用 CSS3 的伪类:before 和:after，分别在被插盒子里面内容的前面和后面动态插入空内容。设置盒子时，每个盒子的大小都要算清楚，小盒子不要超过大盒子范围，而且也不要浪费。使用 z-index 属性设置元素的堆叠顺序，拥有更高堆叠顺序的元素总是会处于堆叠顺序较低的元素的前面，它仅能在定位元素上奏效。

skew()函数能够让元素倾斜显示，它可以将一个对象以其中心位置围绕着 x 轴和 y 轴按照一定的角度倾斜。rotate()函数只是旋转，而不会改变元素的形状。skew()函数不会旋转，而只会改变元素的形状。相关知识将在后面章节介绍。



12.5 案例实战



Note



视频讲解

本节将通过两个案例练习 CSS 盒模型相关组成要素的具体应用。

12.5.1 设计内容页

本例将应用 box-shadow、text-shadow 和 border-radius 等属性，定义一个包含阴影和圆角的特效，同时利用 CSS 渐变、半透明特效设计精致的栏目效果，预览效果如图 12.34 所示。



图 12.34　设计正文内容页

【操作步骤】

第 1 步，新建 HTML5 文档，构建页面结构。

```
<div  class="box">
    <h1>W3C 的“战略漏斗”（Strategy Funnel）：常态化探索创 Web 技术新想法</h1>
    <p>2017 年 6 月 8 日，W3C 的未来战略方向负责人 Wendy Seltzer 发布博客文章，介绍了 W3C 的战略方向意见及研讨交流机制——战略漏斗（Strategy funnel）。</p>
    <p>…</p>
    <p class="right">更多<a href="http://www.chinaw3c.org/archives/1844/" target="_blank">详细内容</a></p>
</div>
```

第 2 步，新建内部样式表，设计页面初始化和包含框的样式。

```
<style type="text/css">
body {/*页面初始化*/
    background-color: #454545;
    margin:1em; padding:0;
}
.box {/*设计包含框样式*/
    border-radius: 10px;                          /*设计圆角*/
    box-shadow: 0 0 12px 1px rgba(205, 205, 205, 1);    /*设计栏目阴影*/
    border: 1px solid black;
    padding: 10px; margin: 24px auto;
    width: 90%;
```



```
        text-shadow: black 1px 2px 2px;    /*设计包含文本阴影*/
        color: white;
        /*设计直线渐变背景*/
        background-image: linear-gradient(to bottom, black, rgba(0, 47, 94, 0.4));
        background-color: rgba(43, 43, 43, 0.5);
    }
```



第 3 步，设计鼠标经过时，放大阴影亮度。

```
.box:hover {box-shadow: 0 0 12px 5px rgba(205, 205, 205, 1);}
```

第 4 步，设计标题样式。在标题正文前面使用 content 生成一个日期图标。

```
h1 {
    font-size: 120%; font-weight: bold;
    text-decoration: underline;
    margin-bottom:34px;
}
/*在标题前面添加额外内容*/
h1:before {content: url(images/date.png); position:relative; top:16px; margin-right:12px;}
```

第 5 步，设计正文段落样式。调整段落文本的行高 、间距，定义字体大小和首行缩进显示。

```
p {
    padding: 6px;
    text-indent:2em;
    line-height:1.8em; font-size:14px;
}
```

12.5.2　设计应用界面



视频讲解

本例利用 CSS3 新增的边框和背景样式来模拟 Windows 7 界面效果，主要应用了 box-shadow、border-radius、text-shadow、border-color、border-image 等属性，同时还用到了渐变设计属性。整个案例的演示效果如图 12.35 所示。

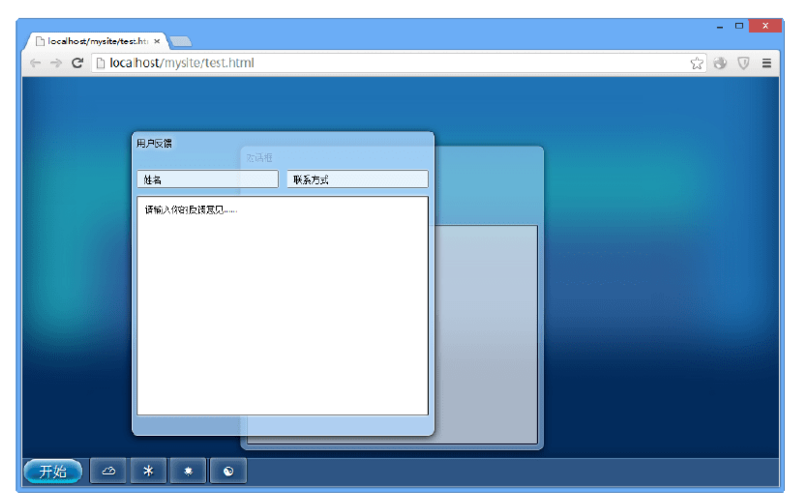


图 12.35　设计 Windows 7 界面效果





Note

【操作步骤】

第 1 步，新建 HTML5 文档，设计页面结构。整个 UI 界面的结构比较简单，说明如下：

```
<div id="desktop">
    <div id="bgWindow" class="window secondary">
        <span>对话框</span>
        <div class="content"></div>
    </div>
    <div id="frontWindow" class="window">
        <span>用户反馈</span>
        <div id="winInput"><input type="text" value="姓名"><input type="text" value="联系方式"></div>
        <div id="winContent" class="content">请输入你的反馈意见……</div>
    </div>
    <div id="startmenu">
        <button id="winflag">开始</button>
        <span id="toolBtn"><!--任务栏图标-->
            <button class="application">☁</button>
            <button class="application">✱</button>
            <button class="application">❋</button>
            <button class="application">☯</button>
        </span>
    </div>
</div>
```

第 2 步，设计桌面效果。在文档样式表中，先定制页面样式，然后设置桌面显示背景，样式代码如下：

```
html,body { /*页面样式定制，清除边距，显式定义高度*/
    padding:0; margin:0;
    height:100%;
}
#desktop { /*定制桌面背景效果*/
    background: #2c609b;
    height:100%;                              /*满窗口显示*/
    font: 12px "Segoe UI", Tahoma, sans-serif;
    position: relative;                       /*定义包含框，为后面的桌面定位元素提供参考*/
    /*定义桌面内阴影，使用一组 3 个内阴影设计梦幻效果*/
    box-shadow: inset 0 -200px 100px #032b5c,
        inset -100px 100px 100px #2073b5,
        inset 100px 200px 100px #1f9bb1;
    overflow: hidden;                         /*隐藏超出的内容*/
}
```

第 3 步，设计开始菜单和任务栏。开始菜单和任务栏主要用到了圆角样式和盒子阴影，在设计任务栏中的图标时，还用到渐变效果，该技术将在后面章节中进行详细说明，该部分的样式代码如下：

```
#startmenu { /*设置任务栏效果*/
    /*固定显示在页面底部*/
    position: absolute; bottom: 0;
    /*固定大小*/
    height: 40px; width: 100%;
    background: rgba(178, 215, 255, 0.25);   /*增加半透明效果*/
```



```
        /*为任务栏设计顶部外阴影，以及在内部添加两道阴影效果*/
        box-shadow: 0 -2px 20px rgba(0, 0, 0, 0.25),
                    inset 0 1px #042754,
                    inset 0 2px #5785b0;
        overflow: hidden;
}
#startmenu button {
        font-size: 1.6em; color: #fff;
        text-shadow: 1px 2px 2px #00294b;              /*为按钮文字增加阴影效果*/
}
#startmenu #winflag { /*设计“开始”按钮样式*/
        float: left;
        margin: 2px; margin-right: 10px;
        height: 34px; width: 80px;
        border: none;
        background: #034a76;
        /*设计“开始”按钮圆角显示*/
        border-radius: 40px;
        /*设计“开始”按钮内外阴影特效*/
        box-shadow: 0 0 1px #fff,
            0 0 3px #000,
             0 0 3px #000,
            inset 0 1px #fff,
            inset 0 12px rgba(255, 255, 255, 0.15),
            inset 0 4px 10px #cef,
            inset 0 22px 5px #0773b4,
            inset 0 -5px 10px #0df;
}
#startmenu .application { /*设计任务栏图标样式*/
        position: relative;
        bottom: 1px; height: 38px; width: 52px;
        background: rgba(14, 59, 103, 0.25);
        border: 1px solid rgba(0, 0, 0, 0.8);
        /*设计渐变特效*/
        transition: .3s all;
        /*设计任务栏图标圆角显示*/
        border-radius: 4px;
        /*设计任务栏图标内外阴影特效*/
        box-shadow: inset 0 0 1px #fff,
            inset 4px 4px 20px rgba(255, 255, 255, 0.33),
            inset -2px -2px 10px rgba(255, 255, 255, 0.25);
}
/*设计鼠标经过时，图标显示为半透明的色彩变化效果*/
#startmenu .application:hover {background-color: rgba(255, 255, 255, 0.25);}
```

第 4 步，设计窗口效果。窗口 UI 主要涉及圆角和半透明效果设计，样式代码如下：

```
/*设计窗口外框效果*/
.window {
        /*定位窗体大小和位置*/
        position: absolute;
        left: 150px; top: 75px;
```





```
        width: 400px; height: 400px; padding: 7px;
        /*设计半透明效果的边框和背景效果*/
        border: 1px solid rgba(255, 255, 255, 0.6);
        background: rgba(178, 215, 255, 0.75);
        /*设计窗体外框圆角显示*/
        border-radius: 8px;
        /*设计窗体外框的外阴影特效*/
        box-shadow: 0 2px 16px #000,
            0 0 1px #000,
            0 0 1px #000;
        /*设计晕边效果*/
        text-shadow: 0 0 15px #fff, 0 0 15px #fff;
    }
    .window span {display: block;}
    .window input { /*文本输入框样式*/
        /*设计文本输入框圆角显示*/
        border-radius: 2px;
        /*设计文本输入框的内外阴影特效*/
        box-shadow: 0 0 2px #fff,
            0 0 1px #fff,
            inset 0 0 3px #fff;
    }
    .window input + input {margin-left: 12px;}
    .window.secondary { /*定位第二个窗体位置和不透明度*/
        left: 300px; top: 125px; opacity: 0.66;
    }
    .window.secondary span {margin-bottom: 85px;}
    .window .content {/*设计窗口内文本区域样式*/
        padding: 10px; height: 279px;
        background: #fff;    border: 1px solid #000;
        /*设计文本区域圆角显示*/
        border-radius: 2px;
        /*设计文本区域的内外阴影特效*/
        box-shadow: 0 0 5px #fff,
            0 0 1px #fff,
            inset 0 1px 2px #aaa;
        text-shadow: none; /*取消文本阴影*/
    }
```

12.6 在线练习

本节专题练习 CSS3 的用户界面特性，感兴趣的读者可以扫码练习。



在线练习



设计 CSS3 动画

CSS3 动画包括过渡动画和关键帧动画，它们主要通过改变 CSS 属性值来模拟实现。本章将详细介绍 Transform、Transitions 和 Animations 三大功能模块，其中 Transform 实现对网页对象的变形操作，Transitions 实现 CSS 属性过渡变化，Animations 实现 CSS 样式分步式演示效果。

【学习重点】

- 设计对象变形操作。
- 设计过渡样式。
- 设计关键帧动画。
- 能够灵活使用 CSS3 动画设计页面特效。



13.1 CSS3 变形



Note

2012 年 9 月，W3C 发布了 CSS3 变形工作草案。CSS3 变形允许 CSS 把元素转变为 2D 或 3D 空间，这个草案包括 CSS3 2D 变形和 CSS3 3D 变形。

权威参考：http://www.w3.org/TR/css-transforms-1/。

权威参考

13.1.1 认识 Transform

CSS3 变形是多种效果的集合，如旋转、缩放、平移和倾斜等，每个效果都被称为变形函数，它们可以操控元素发生旋转、缩放、平移和倾斜等变化。在 CSS3 之前，实现类似的效果需要图片、Flash 或 JavaScript 才能完成。而使用 CSS 来完成这些变形则无须加载额外的文件，提高了开发效率和页面的执行效率。

CSS3 变形包括 3D 变形和 2D 变形，3D 变形使用基于 2D 变形的相同属性，如果了解了 2D 变形，会发现 3D 变形与 2D 变形的功能类似。

CSS 2D Transform 获得了各主流浏览器的支持，但是 CSS 3D Transform 支持程度不是很完善。考虑到浏览器兼容性，3D 变形在实际应用时应添加私有属性，并且个别属性在某些主流浏览器中并未得到很好的支持，简单说明如下：

- ☑ 在 IE10+中，3D 变形部分属性未得到很好的支持。
- ☑ Firefox10.0 至 Firefox15.0 版本的浏览器，在使用 3D 变形时需要添加私有属性-moz-，但从 Firefox16.0+版本开始无须添加浏览器私有属性。
- ☑ Chrome12.0+版本中使用 3D 变形时需要添加私有属性-webkit-。
- ☑ Safari4.0+版本中使用 3D 变形时需要添加私有属性-webkit-。
- ☑ Opera15.0+版本才开始支持 3D 变形，使用时需要添加私有属性-webkit-。
- ☑ 移动设备中 iOS Safari3.2+、Android Browser3.0+、Blackberry Browser7.0+、Opera Mobile24.0+、Chrome for Android25.0+都支持 3D 变形，但在使用时需要添加私有属性-webkit-；Firefox for Android19.0+支持 3D 变形，且无须添加浏览器私有属性。



视频讲解

13.1.2 设置原点

CSS 变形的原点默认为对象的中心点（50% 50%），使用 transform-origin 属性可以重新设置变形原点。语法格式如下：

```
transform-origin: [<percentage> | <length> | left | center① | right] [<percentage> | <length> | top | center② | bottom]?
```

取值简单说明如下。

- ☑ <percentage>：用百分比指定坐标值。可以为负值。
- ☑ <length>：用长度值指定坐标值。可以为负值。
- ☑ left：指定原点的横坐标为 left。
- ☑ center①：指定原点的横坐标为 center。
- ☑ right：指定原点的横坐标为 right。
- ☑ top：指定原点的纵坐标为 top。





☑　center②：指定原点的纵坐标为 center。

☑　bottom：指定原点的纵坐标为 bottom。

【示例】通过重置变形原点，可以设计不同的变形效果。在下面示例中以图像的右上角为原点逆时针旋转图像 45°，则比较效果如图 13.1 所示。

```
<style type="text/css">
img {/*固定两幅图像相同大小和相同显示位置*/
    position: absolute;
    left: 20px;
    top: 10px;
    width: 170px;
    width: 250px;
}
img.bg {/*设置第 1 幅图像作为参考*/
    opacity: 0.3;
    border: dashed 1px red;
}
img.change {/*变形第 2 幅图像*/
    border: solid 1px red;
    transform-origin: top right;        /*以右上角为原点进行变形*/
    transform: rotate(-45deg);          /*逆时针旋转 45° */
}
</style>
<img class="bg" src="images/1.jpg">
<img class="change" src="images/1.jpg">
```

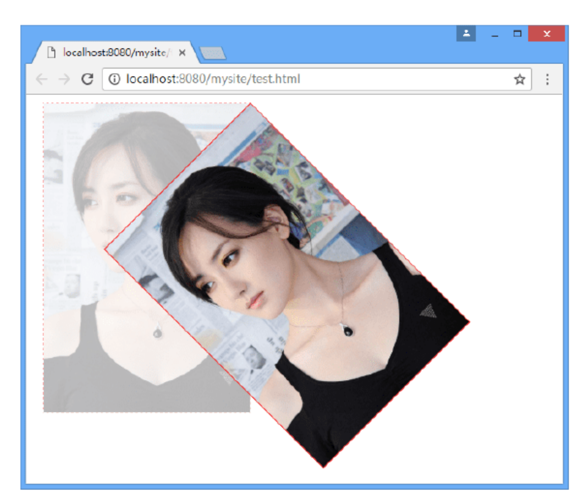


图 13.1　自定义旋转原点

13.1.3　2D 旋转



rotate()函数能够在 2D 空间内旋转对象，语法格式如下：

```
rotate(<angle>)
```

其中参数 angle 表示角度值，取值单位可以是：度，如 90deg（90 度，一圈 360 度）；梯度，如



100grad（相当于 90 度，360 度等于 400grad）；弧度，如 1.57rad（约等于 90 度，360 度等于 2π）；圈，如 0.25turn（等于 90 度，360 度等于 1turn）。



Note

【示例】以 13.1.2 节示例为基础，下面按默认原点逆时针旋转图像 45°，效果如图 13.2 所示。

```
img.change {
    border: solid 1px red;
    transform: rotate(-45deg);
}
```

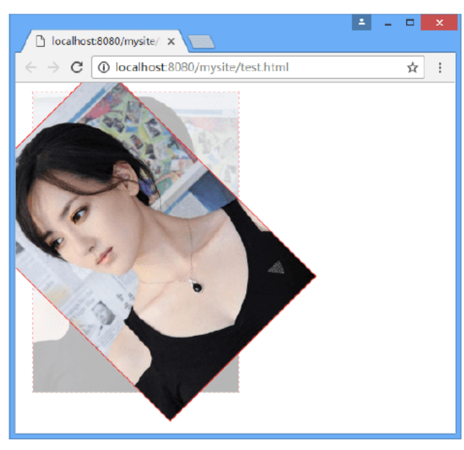


图 13.2　定义旋转效果



视 频 讲 解

13.1.4　2D 缩放

scale()函数能够缩放对象大小，语法格式如下：

```
scale(<number>[, <number>])
```

该函数包含两个参数值，分别用来定义宽和高缩放比例，取值简单说明如下。

☑　如果取值为正数，则基于指定的宽度和高度放大或缩小对象。

☑　如果取值为负数，则不会缩小元素，而是翻转元素（如文字被反转），然后再缩放元素。

☑　如果取值为小于 1 的小数（如 0.5），可以缩小元素。

☑　如果第二个参数省略，则第二个参数等于第一个参数值。

【示例】以 13.1.2 节示例为基础，下面按默认原点把图像缩小为原来的二分之一，效果如图 13.3 所示。

```
img.change {
    border: solid 1px red;
    transform: scale(0.5);
}
```



视 频 讲 解

13.1.5　2D 平移

translate()函数能够平移对象的位置，语法格式如下：

```
translate(<translation-value>[, <translation-value>])
```



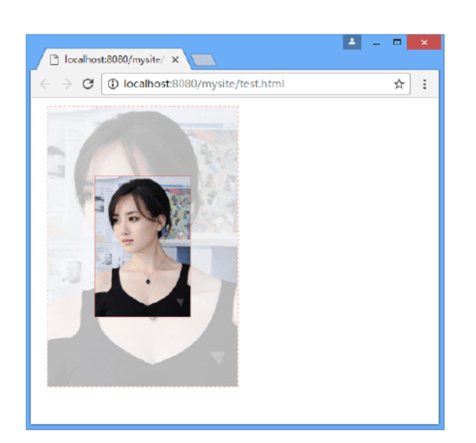


图 13.3　缩小对象效果



该函数包含两个参数值，分别用来定义对象在 X 轴和 Y 轴相对于原点的偏移距离。如果省略参数，则默认值为 0。如果取负值，则表示反向偏移，参考原点保持不变。

【示例】 下面示例设计向右下角方向平移图像，其中 X 轴偏移 150px，Y 轴偏移 50px，演示效果如图 13.4 所示。

```
img.change {
    border: solid 1px red;
    transform: translate(150px, 50px);
}
```

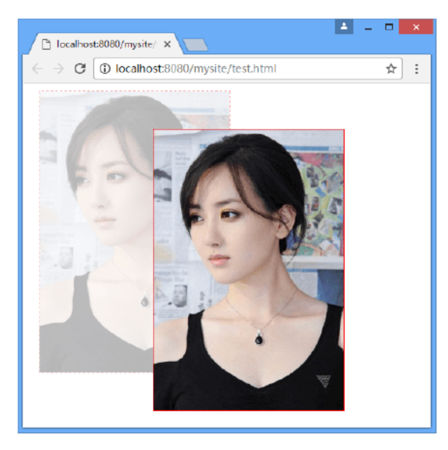


图 13.4　平移对象效果

13.1.6　2D 倾斜



视频讲解

skew()函数能够倾斜显示对象，语法格式如下：

```
skew(<angle> [, <angle>])
```

该函数包含两个参数值，分别用来定义对象在 X 轴和 Y 轴倾斜的角度。如果省略参数，则默认值为 0。与 rotate()函数不同，rotate()函数只是旋转对象的角度，而不会改变对象的形状；skew()函数会改变对象的形状。



【示例】下面示例使用 skew()函数变形图像，X 轴倾斜 30°，Y 轴倾斜 20°，效果如图 13.5 所示。



```
img.change {
    border: solid 1px red;
    transform: skew(30deg, 20deg);
}
```

Note

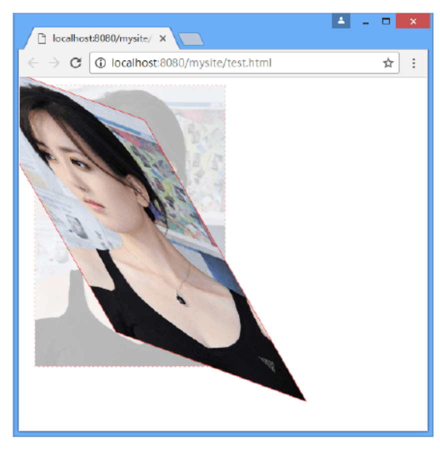


图 13.5　倾斜对象效果



视频讲解

13.1.7　2D 矩阵

matrix()是一个矩阵函数，它可以同时实现缩放、旋转、平移和倾斜操作，语法格式如下：

```
matrix(<number>, <number>, <number>, <number>, <number>, <number>)
```

该函数包含 6 个值，具体说明如下：

☑　第 1 个参数控制 X 轴缩放。
☑　第 2 个参数控制 X 轴倾斜。
☑　第 3 个参数控制 Y 轴倾斜。
☑　第 4 个参数控制 Y 轴缩放。
☑　第 5 个参数控制 X 轴平移。
☑　第 6 个参数控制 Y 轴平移。

【示例】下面示例使用 matrix()函数模拟 13.1.6 节示例的倾斜变形操作，效果类似图 13.5 所示。

```
img.change {
    border: solid 1px red;
    transform: matrix(1, 0.6, 0.2, 1, 0, 0);
}
```

【补充】

多个变形函数可以在一个声明中同时定义。例如：

```
div {
    transform: translate(80, 80);
    transform: rotate(45deg);
    transform: scale(1.5, 1.5);
}
```



针对上面样式，可以简化为：

```
div {transform: translate(80, 80) rotate(45deg) scale(1.5, 1.5);}
```

13.1.8 设置变形类型



Note



视频讲解

CSS3 变形包括 2D 和 3D 两种类型，使用 transform-style 属性可以设置 CSS 变形的类型，语法格式如下：

```
transform-style: flat | preserve-3d
```

取值简单说明如下。

☑ flat：指定子元素在该元素所在平面内进行变形，即 2D 平面变形，为默认值。

☑ preserve-3d：指定子元素定位在三维空间内进行变形，即 3D 立体变形。

【示例】 借助上面示例，下面示例使用<div id="box">容器包裹两幅图像，改进后的 HTML 结构如下：

```
<div id="box">
    <img class="bg" src="images/1.jpg">
    <img class="change" src="images/1.jpg">
</div>
```

为<div id="box">容器设置 CSS3 变形类型为 3D，样式代码如下：

```
#box {
    transform-style: preserve-3d;
}
```

为 change 图像应用 3D 顺时针旋转 45°，CSS 样式如下：

```
img.change {
    border: solid 1px red;
    transform: translate3d(60px, 60px, 400px);
}
```

在浏览器中预览，效果如图 13.6 所示。

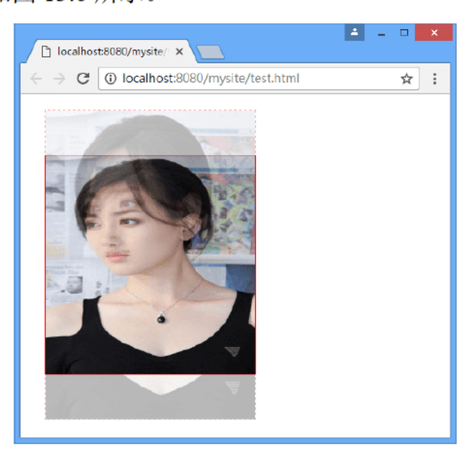
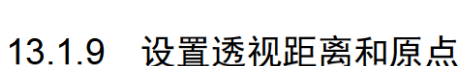


图 13.6　3D 平移效果


视频讲解


Note



13.1.9 设置透视距离和原点

3D 变形与 2D 变形最大的不同就在于其参考的坐标轴不同：2D 变形的坐标轴是平面的，只存在 X 轴和 Y 轴，而 3D 变形的坐标轴则是 X、Y、Z 轴组成的立体空间，X 轴正向、Y 轴正向、Z 轴正向分别朝向右、下和屏幕外，示意如图 13.7 所示。

透视是 3D 变形中最重要的概念。如果不设置透视，元素的 3D 变形效果将无法实现。在 13.1.8 节示例中，如果使用函数 rotateX(45deg)将图像以 X 轴方向为轴沿顺时针旋转 45°，由于没有设置透视样式的效果，可以看到浏览器将图像的 3D 变形操作垂直投射到 2D 视平面上，最终呈现出来的只是图像的宽高变化。

【示例 1】在 13.1.8 节示例基础上，在<div id="box">容器外设置透视点距离为 1200px，样式代码如下：

```
body{
    perspective: 1200px;
}
```

在浏览器中可以看到如图 13.8 所示的变形效果。

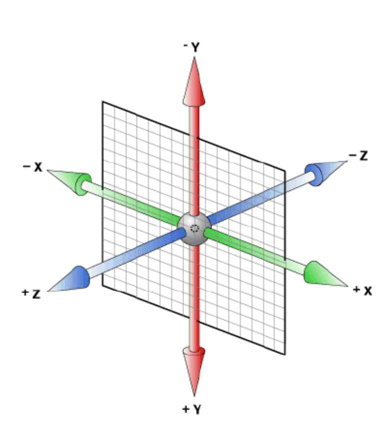


图 13.7　3D 坐标轴示意图

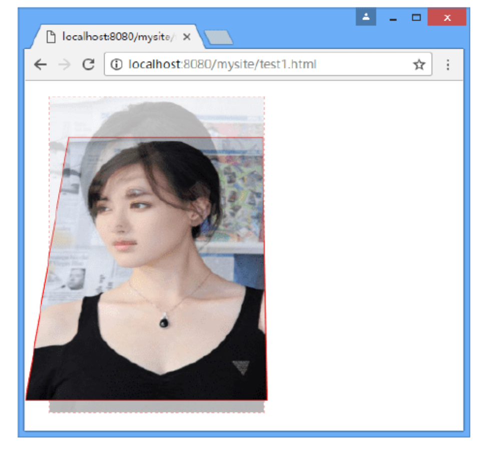


图 13.8　沿 X 轴 3D 旋转 45°效果图

基于对上面示例的直观体验，下面来了解几个核心概念：变形元素、观察者和被透视元素，三者关系如图 13.9 所示。

☑ 变形元素：需要进行 3D 变形的元素。主要设置 transform、transform-origin、backface-visibility 等属性。
☑ 观察者：浏览器模拟出来的用来观察被透视元素的一个没有尺寸的点，观察者发出视线，类似于一个点光源发出光线。
☑ 被透视元素：被观察者观察的元素，根据属性设置的不同，有可能是变形对象本身，也可能是它的父级或祖先元素，主要设置 perspective、perspective-origin 等属性。

1．透视距离

透视距离是指观察者沿着平行于 Z 轴的视线与屏幕之间的距离，也称为视距，示意图如图 13.10 所示。





图 13.9　变形元素、观察者和被透视元素位置关系示意图

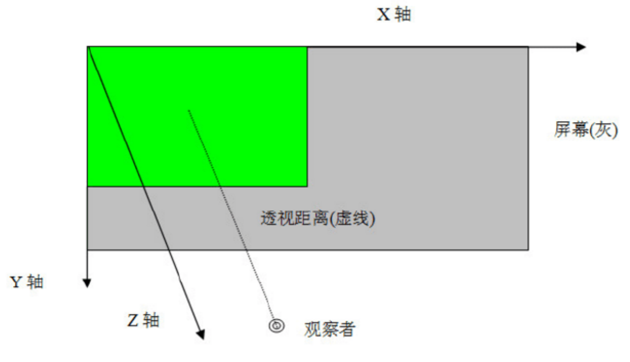


图 13.10　透视距离示意图

使用 perspective 属性可以定义透视距离，语法格式如下：

```
perspective: none | <length>
```

取值简单说明如下。

☑　none：不指定透视。

☑　<length>：指定观察者距离平面的距离，为元素及其内容应用透视变换。

注意： 透视距离不可为 0 和负数，因为观察者与屏幕距离为 0 时或者在屏幕背面时是不可以观察到被透视元素的正面的。perspective 也不可取百分比值，因为百分比值需要相对的元素，但 Z 轴并没有可相对的元素尺寸。

一般地，物体离得越远，显得越小。反映在 perspective 属性上，就是该属性值越大，元素的 3D 变形效果越不明显。

设置 perspective 属性的元素就是被透视元素。一般地，该属性只能设置在变形元素的父级或祖先级。因为浏览器会为其子级的变形产生透视效果，但并不会为其自身产生透视效果。应用示例可以参考上面示例 1。





2．透视原点

透视原点是指观察者的位置，一般观察者位于与屏幕平行的另一个平面上，观察者始终是与屏幕垂直的。观察者的活动区域是被观察元素的盒模型区域，示意图如图 13.11 所示。

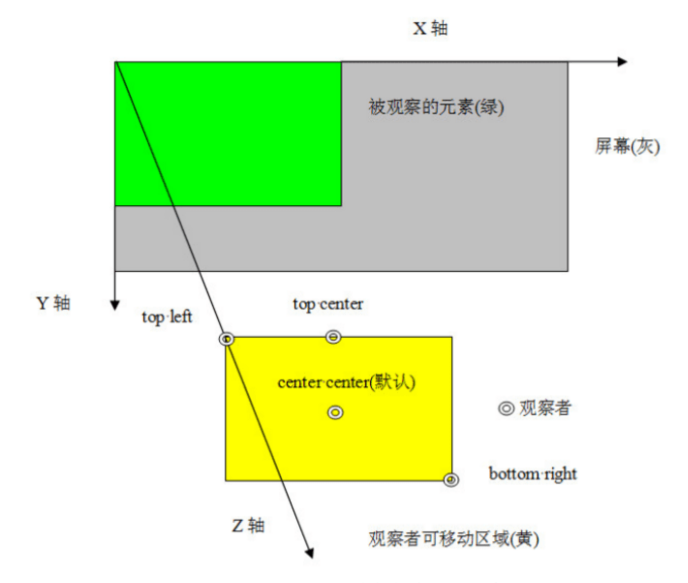


图 13.11　透视原点的位置区域

使用 perspective-origin 属性可以定义透视点的位置，语法格式如下：

```
perspective-origin: [<percentage> | <length> | left | center① | right] [<percentage> | <length> | top | center② | bottom]?
```

取值简单说明如下。

☑　<percentage>：用百分比指定透视点坐标值，相对于元素宽度。可以为负值。

☑　<length>：用长度值指定透视点坐标值。可以为负值。

☑　left：指定透视点的横坐标为 left。

☑　center①：指定透视点的横坐标为 center。

☑　right：指定透视点的横坐标为 right。

☑　top：指定透视点的纵坐标为 top。

☑　center②：指定透视点的纵坐标为 center。

☑　bottom：指定透视点的纵坐标为 bottom。

【示例 2】在示例 1 基础上，设置观察点在右侧居中位置，则显示效果如图 13.12 所示。

```
body{
    perspective: 1200px;
    perspective-origin: right;
}
```



视频讲解

13.1.10　3D 平移

3D 平移主要包括下面 4 个函数。

☑　translatex(<translation-value>)：指定对象 X 轴（水平方向）的平移。



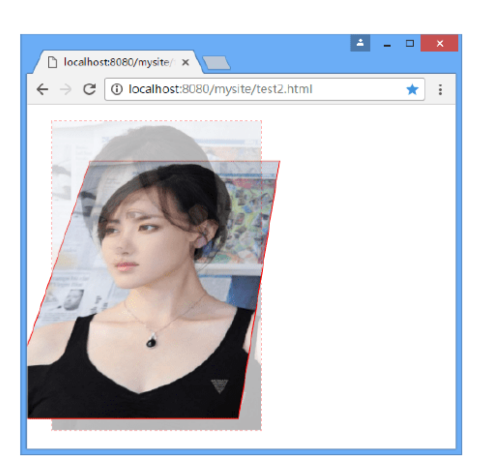


图 13.12　设置观察点位置在右侧效果



- ☑ translatey(<translation-value>)：指定对象 Y 轴（垂直方向）的平移。
- ☑ translatez(<length>)：指定对象 Z 轴的平移。
- ☑ translate3d(<translation-value>,<translation-value>,<length>)：指定对象的 3D 平移。第 1 个参数对应 X 轴，第 2 个参数对应 Y 轴，第 3 个参数对应 Z 轴，参数不允许省略。

参数<translation-value>表示<length>或<percentage>，即 X 轴和 Y 轴可以取值长度值或百分比，但是 Z 轴只能够设置长度值。

【示例】下面示例设计图像在 3D 空间中平移，形成一种错位效果，如图 13.13 所示。

```
#box {
    transform-style: preserve-3d;
    perspective: 1200px;
}
img.change {
    border: solid 1px red;
    transform: translate3d(200px, 30px, 60px);
}
```

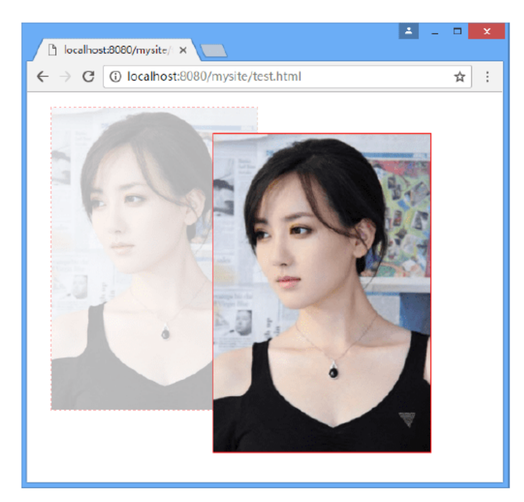


图 13.13　定义 3D 平移效果



从图 13.13 效果可以看出，当 Z 轴值越大时，元素离浏览者越近，从视觉上元素就变得更大；反之，其值越小时，元素也离观看者越远，从视觉上元素就变得更小。



提示： translateZ()函数在实际使用中等效于 translate3d(0,0,tz)。仅从视觉效果上看，translateZ()和 translate3d(0,0,tz)函数功能非常类似于二维空间的 scale()缩放函数，但实际上完全不同。translateZ()和 translate3d(0,0,tz)变形是发生在 Z 轴上，而不是在 X 轴或 Y 轴。



视频讲解

13.1.11　3D 缩放

3D 缩放主要包括下面 4 个函数。

☑　scalex(<number>)：指定对象 X 轴的（水平方向）缩放。

☑　scaley(<number>)：指定对象 Y 轴的（垂直方向）缩放。

☑　scalez(<number>)：指定对象的 Z 轴缩放。

☑　scale3d(<number>,<number>,<number>)：指定对象的 3D 缩放。第 1 个参数对应 X 轴，第 2 个参数对应 Y 轴，第 3 个参数对应 Z 轴，参数不允许省略。

参数<number>为一个数字，表示缩放倍数，可参考 2D 缩放参数说明。

【示例】下面以 13.1.10 节示例为基础，在 X 轴和 Y 轴放大图像 1.5 倍，Z 轴放大图像 2 倍，然后使用 translatex()把变形的图像移到右侧显示，以便与原图进行比较，演示效果如图 13.14 所示。

```
img.change {
    border: solid 1px red;
    transform: scale3D(1.5,1.5,2) translatex(240px);
}
```

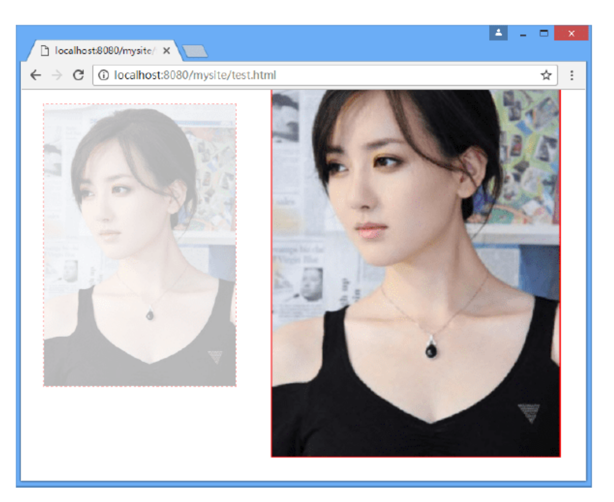


图 13.14　定义 3D 缩放效果



视频讲解

13.1.12　3D 旋转

3D 旋转主要包括下面 4 个函数。

☑　rotatex(<angle>)：指定对象在 X 轴上的旋转角度。

☑　rotatey(<angle>)：指定对象在 Y 轴上的旋转角度。



☑　rotatez(<angle>)：指定对象在 Z 轴上的旋转角度。

☑　rotate3d(<number>,<number>,<number>,<angle>)：指定对象的 3D 旋转角度，其中前 3 个参数分别表示旋转的方向 X、Y、Z，第 4 个参数表示旋转的角度，参数不允许省略。



提示： rotate3d()函数前 3 个参数值分别用来描述围绕 X、Y、Z 轴旋转的矢量值。最终变形元素以由(0,0,0)和(x,y,z)这两个点构成的直线为轴，进行旋转。当第 4 个参数为正数时，元素进行顺时针旋转；当第 4 个参数为负数时，元素进行逆时针旋转。

rotate3d()函数可以与前面 3 个旋转函数进行转换，简单说明如下：

☑　rotatex(a)函数功能等同于 rotate3d(1,0,0,a)。

☑　rotatey(a)函数功能等同于 rotate3d(0,1,0,a)。

☑　rotatez(a)函数功能等同于 rotate3d(0,0,1,a)。

【示例】 以 13.1.1 节示例为基础，使用 rotate3d()函数顺时针旋转图像 45°，其中 X 轴、Y 轴和 Z 轴数值分别为 2、2、1，效果如图 13.15 所示。

```
img.change {
    border: solid 1px red;
    transform: rotate3d(2,2,1,45deg);
}
```

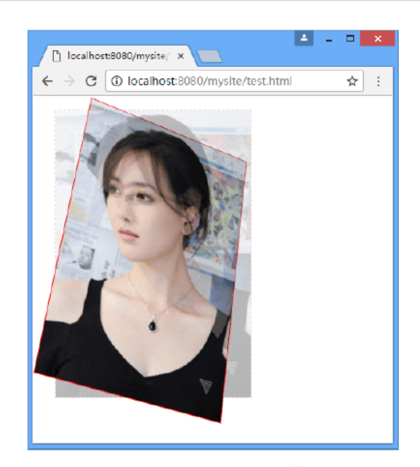


图 13.15　定义 3D 旋转效果

13.1.13　透视函数



perspective 属性可以定义透视距离，它应用在变形元素的父级或祖先级元素上。而透视函数 perspective()是 transform 变形函数的一个属性值，可以应用于变形元素本身。具体语法格式如下：

```
perspective(<length>)
```

参数是一个长度值，值只能是正数。

【示例】 下面示例设计图像在 X 轴上旋转 120°，透视距离为 180px，如图 13.16 所示。

```
#box {transform-style: preserve-3d;}
img.change {
    border: solid 1px red;
```



```
        transform:perspective(180px) rotateX(120deg);
    }
```



Note

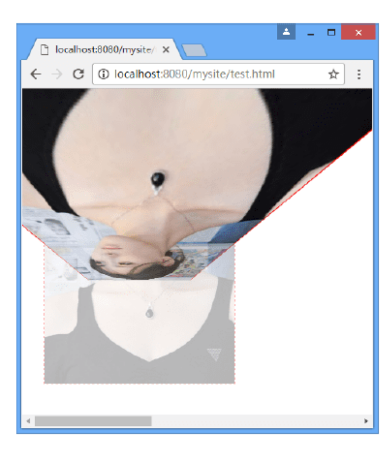


图 13.16　定义 3D 旋转效果

注意：由于 transform 属性是从前向后的顺序解析属性值的，所以一定要把 perspective()函数写在其他变形函数前面，否则将没有透视效果。

由于透视原点 perspective-origin 只能设置在设置了 perspective 透视属性的元素上，若为元素设置透视函数 perspective()，则透视原点不起作用，观察者使用默认位置，即元素中心点对应的平面上。

13.1.14　变形原点

2D 变形原点由于没有 Z 轴，所以 Z 轴的值默认为 0。在 3D 变形原点中，Z 轴是一个可以设置的变量。语法格式如下：

```
transform-origin: X 轴  Y 轴  Z 轴
```

取值简单说明如下。

☑　X 轴：left | center | right | <length> | <percentage>。

☑　Y 轴：top | center | bottom | <length> | <percentage>。

☑　Z 轴：<length>。

对于 X 轴和 Y 轴来说，可以设置关键字和百分比，分别相对于其本身元素水平方向的宽度和垂直方向的高度和；Z 轴只能设置长度值。



视频讲解

13.1.15　背景可见

元素的背面在默认情况下是可见的，有时可能需要让元素背面不可见，这时就可以使用 backface-visibility 属性，该属性的具体语法格式如下：

```
backface-visibility: visible | hidden
```

取值简单说明如下。

☑　visible：指定元素背面可见，允许显示正面的镜像，为默认值。



☑　hidden：指定元素背面不可见。

【示例】以 13.1.13 节示例为基础，如果在变形图像样式中添加“backface-visibility: hidden;”，定义元素背面面向用户时不可见，这时再次预览，则会发现变形图像已经不存在，因为它的背面面向用户，被隐藏了，效果如图 13.17 所示。

```
img.change {
    border: solid 1px red;
    transform:perspective(180px) rotateX(120deg);
    backface-visibility: hidden;
}
```

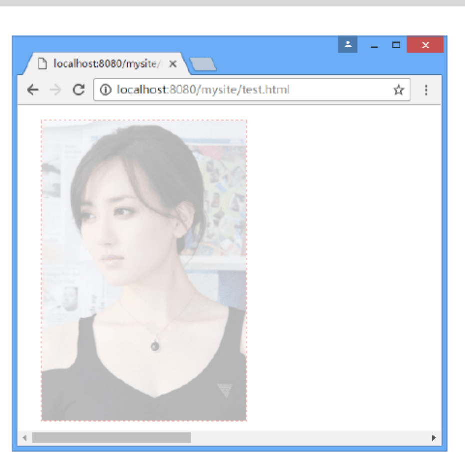


图 13.17　定义背面面向用户不可见效果

13.2　过 渡 动 画

2013 年 2 月，W3C 发布了 CSS Transitions 工作草案，在这个草案中描述了 CSS 过渡动画的基本实现方法和属性。目前获得所有浏览器的支持，包括支持带有前缀（私有属性）或不带前缀（标准属性）的过渡。最新版本浏览器（IE 10+、Firefox 16+和 Opera 12.5+）均支持不带前缀的过渡属性 transition，而旧版浏览器则支持带前缀的过渡，如 Webkit 引擎支持-webkit-transition 私有属性，Mozilla Gecko 引擎支持-moz-transition 私有属性，Presto 引擎支持-o-transition 私有属性，IE6～IE9 浏览器不支持 transition 属性，IE10 支持 transition 属性。

权威参考：http://www.w3.org/TR/css3-transitions/。

权威参考　视频讲解

13.2.1　设置过渡属性

transition-property 属性用来定义过渡动画的 CSS 属性名称，基本语法如下：

```
transition-property: none | all | [<IDENT>] [',' <IDENT>]*;
```

取值简单说明如下。

☑　none：表示没有元素。

☑　all：默认值，表示针对所有元素，包括:before 和:after 伪元素。





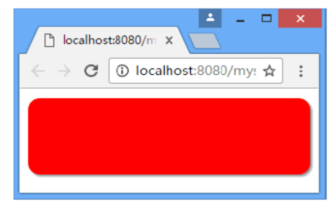
Note

☑ IDENT：指定 CSS 属性列表。几乎所有色彩、大小或位置等相关的 CSS 属性，包括许多新添加的 CSS3 属性，都可以应用过渡，如 CSS3 变换中的放大、缩小、旋转、斜切、渐变等。

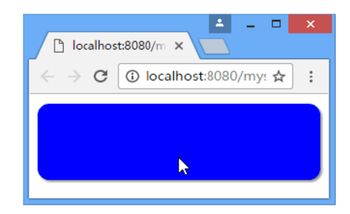
【示例】在下面示例中，指定动画的属性为背景颜色。这样当鼠标经过盒子时，会自动从红色背景过渡到蓝色背景，演示效果如图 13.18 所示。

```
<style type="text/css">
div {
    margin: 10px auto; height: 80px;
    background: red;
    border-radius: 12px;
    box-shadow: 2px 2px 2px #999;
}
div:hover {
    background-color: blue;
    /*指定动画过渡的 CSS 属性*/
    transition-property: background-color;
}
</style>

<div></div>
```



默认状态



鼠标经过时被旋转

图 13.18　定义简单的背景色切换动画



视 频 讲 解

13.2.2　设置过渡时间

transition-duration 属性用来定义转换动画的时间长度，基本语法如下：

```
transition-duration:<time> [, <time>]*;
```

初始值为 0，适用于所有元素，包括:before 和:after 伪元素。在默认情况下，动画过渡时间为 0s，所以当指定元素动画时，会看不到过渡的过程，直接看到结果。

【示例】以 13.2.1 节示例为例，下面示例设置动画过渡时间为 2s，当鼠标移过对象时，会看到背景色从红色逐渐过渡到蓝色，演示效果如图 13.19 所示。

```
div:hover {
    background-color: blue;
    /*指定动画过渡的 CSS 属性*/
    transition-property: background-color;
    /*指定动画过渡的时间*/
    transition-duration:2s;
}
```



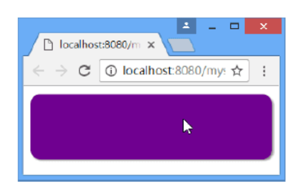


图 13.19　设置动画时间



Note



视 频 讲 解

13.2.3　设置延迟过渡时间

transition-delay 属性用来定义开启过渡动画的延迟时间，基本语法如下：

```
transition-delay:<time> [, <time>]*;
```

初始值为 0，适用于所有元素，包括:before 和:after 伪元素。设置时间可以为正整数、负整数和零，非零时必须设置单位是 s（秒）或者 ms（毫秒），为负数时，过渡的动作会从该时间点开始显示，之前的动作被截断；为正数时，过渡的动作会延迟触发。

【示例】继续以 13.2.2 节示例为基础进行介绍，下面示例设置过渡动画推迟 2s 后执行，则当鼠标移过对象时，会看不到任何变化，2s 之后，才发现背景色从红色逐渐过渡到蓝色。

```
div:hover {
    background-color: blue;
    /*指定动画过渡的 CSS 属性*/
    transition-property: background-color;
    /*指定动画过渡的时间*/
    transition-duration: 2s;
    /*指定动画延迟触发*/
    transition-delay: 2s;
}
```

视 频 讲 解

13.2.4　设置过渡动画类型

transition-timing-function 属性用来定义过渡动画的类型，基本语法如下：

```
transition-timing-function:ease | linear | ease-in | ease-out | ease-in-out | cubic-bezier(<number>, <number>, <number>, <number>) [, ease | linear | ease-in | ease-out | ease-in-out | cubic-bezier(<number>, <number>,<number>, <number>)]*
```

属性初始值为 ease，取值简单说明如下。

☑　ease：平滑过渡，等同于 cubic-bezier(0.25, 0.1, 0.25, 1.0)函数，即立方贝塞尔曲线效果。

☑　linear：线性过渡，等同于 cubic-bezier(0.0, 0.0, 1.0, 1.0)函数。

☑　ease-in：由慢到快，等同于 cubic-bezier(0.42, 0, 1.0, 1.0)函数。

☑　ease-out：由快到慢，等同于 cubic-bezier(0, 0, 0.58, 1.0)函数。

☑　ease-in-out：由慢到快再到慢，等同于 cubic-bezier(0.42, 0, 0.58, 1.0)函数。

☑　cubic-bezier：特殊的立方贝塞尔曲线效果。

【示例】以 13.2.2 节示例为基础进行介绍，下面设置过渡类型为线性效果，代码如下：

```
div:hover {
    background-color: blue;
    /*指定动画过渡的 CSS 属性*/
```



```
        transition-property: background-color;
        /*指定动画过渡的时间*/
        transition-duration: 10s;
        /*指定动画过渡为线性效果*/
        transition-timing-function: linear;
}
```



Note



视频讲解

13.2.5 设置过渡触发动作

CSS3 过渡动画一般通过动态伪类触发，如表 13.1 所示。也可以通过 JavaScript 事件触发，包括 click、focus、mousemove、mouseover、mouseout 等。

表 13.1 CSS 动态伪类

动 态 伪 类	作 用 元 素	说　明
:link	只有链接	未访问的链接
:visited	只有链接	访问过的链接
:hover	所有元素	鼠标经过元素
:active	所有元素	鼠标单击元素
:focus	所有可被选中的元素	元素被选中

1．:hover

最常用的过渡触发方式是使用:hover 伪类。

【示例 1】 下面示例设计当鼠标经过 div 元素时，该元素的背景颜色在经过 1s 的初始延迟后，于 2s 内动态地从绿色变为蓝色。

```
<style type="text/css">
div {
        margin: 10px auto;
        height: 80px;
        border-radius: 12px;
        box-shadow: 2px 2px 2px #999;
        background-color: red;
        transition: background-color 2s ease-in 1s;
}
div:hover {background-color: blue}
</style>
<div></div>
```

2．:active

:active 伪类表示用户单击某个元素并按住鼠标按键时显示的状态。

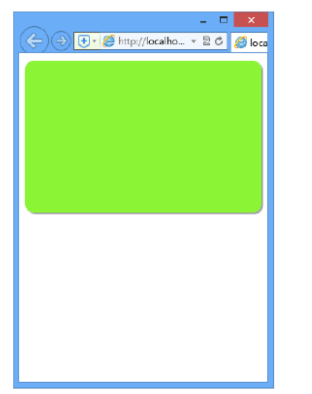
【示例 2】 下面示例设计当用户单击 div 元素时，该元素被激活，这时会触发动画，高度属性从 200px 过渡到 400px。如果按住该元素，保持住活动状态，则 div 元素始终显示 400px 高度，松开鼠标之后，又会恢复原来的高度，如图 13.20 所示。

```
<style type="text/css">
div {
```

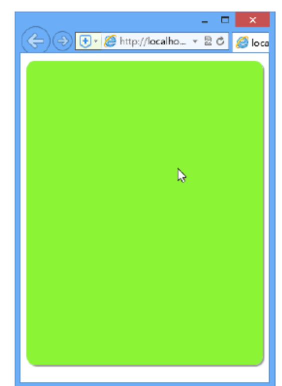





```
        margin: 10px auto;
        border-radius: 12px;
        box-shadow: 2px 2px 2px #999;
        background-color: #8AF435;
        height: 200px;
        transition: width 2s ease-in;
}
div:active {height: 400px;}
</style>
<div></div>
```



默认状态



单击

图 13.20　定义激活触发动画

3．: focus

:focus 伪类通常会在表单对象接收键盘响应时出现。

【示例 3】下面设计当输入框获取焦点时，输入框的背景色逐步高亮显示，如图 13.21 所示。

```
<style type="text/css">
label {
        display: block;
        margin: 6px 2px;
}
input[type="text"], input[type="password"] {
        padding: 4px;
        border: solid 1px #ddd;
        transition: background-color 1s ease-in;
}
input:focus {background-color: #9FFC54;}
</style>
<form id=fm-form action="" method=post>
        <fieldset>
                <legend>用户登录</legend>
                <label for="name">姓名
                        <input type="text" id="name" name="name" >
                </label>
                <label for="pass">密码
```



```
            <input type="password" id="pass" name="pass" >
        </label>
    </fieldset>
</form>
```



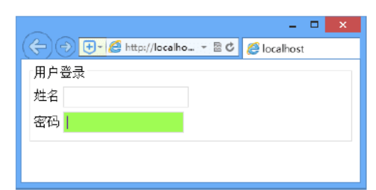


图 13.21　定义获取焦点触发动画

提示：把:hover 伪类与:focus 配合使用，能够丰富鼠标用户和键盘用户的体验。

4．:checked

:checked 伪类在发生选中状况时触发过渡，取消选中则恢复原来状态。

【示例 4】下面示例设计当复选框被选中时缓慢缩进两个字符，演示效果如图 13.22 所示。

```
<style type="text/css">
label.name {
    display: block;
    margin: 6px 2px;
}
input[type="text"], input[type="password"] {
    padding: 4px;
    border: solid 1px #ddd;
}
input[type="checkbox"] {transition: margin 1s ease;}
input[type="checkbox"]:checked {margin-left: 2em;}
</style>
<form id=fm-form action="" method=post>
    <fieldset>
        <legend>用户登录</legend>
        <label class="name" for="name">姓名
            <input type="text" id="name" name="name" >
        </label>
        <p>技术专长<br>
            <label>
                <input type="checkbox" name="web" value="html" id="web_0">
                HTML</label><br>
            <label>
                <input type="checkbox" name="web" value="css" id="web_1">
                CSS</label><br>
            <label>
                <input type="checkbox" name="web" value="javascript" id="web_2">
                JavaScript</label><br>
        </p>
    </fieldset>
</form>
```



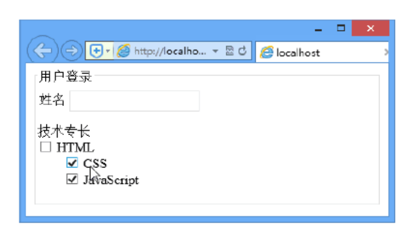


图 13.22　定义被选中时触发动画

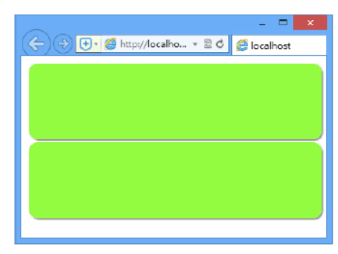


5．媒体查询

触发元素状态变化的另一种方法是使用 CSS3 媒体查询，媒体查询的相关知识在第 14 章详细介绍。

【示例 5】下面示例设计 div 元素的宽度和高度为 49%，200px，如果用户将窗口大小调整到 420px 或以下，则该元素将过渡为 100%，100px。也就是说，当窗口宽度变化经过 420px 的阈值时，将会触发过渡动画，如图 13.23 所示。

```
<style type="text/css">
div {
    float: left; margin: 2px;
    width: 49%; height: 200px;
    background: #93FB40;
    border-radius: 12px;
    box-shadow: 2px 2px 2px #999;
    transition: width 1s ease, height 1s ease;
}
@media only screen and (max-width: 420px) {
    div {
        width: 100%;
        height: 100px;
    }
}
</style>
<div></div>
<div></div>
```



窗口宽度小于等于 420px



窗口宽度大于 420px

图 13.23　媒体查询触发动画

如果网页加载时用户的窗口大小是 420px 或以下，浏览器会在该部分应用这些样式，但是由于不会出现状态变化，因此不会发生过渡。





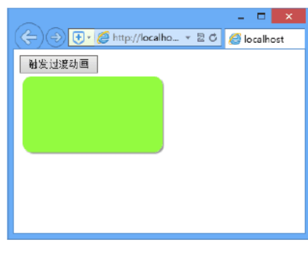
Note

6．JavaScript 事件

【示例 6】下面示例可以使用纯粹的 CSS 伪类触发过渡，为了方便用户理解，这里通过 jQuery 脚本触发过渡。

```
<script type="text/javascript" src="images/jquery-1.10.2.js"></script>
<script type="text/javascript">
$(function() {
    $("#button").click(function() {
        $(".box").toggleClass("change");
    });
});
</script>
<style type="text/css">
.box {
    margin:4px;
    background: #93FB40;
    border-radius: 12px;
    box-shadow: 2px 2px 2px #999;
    width: 50%; height: 100px;
    transition: width 2s ease, height 2s ease;
}
.change { width: 100%; height: 120px;}
</style>
<input type="button" id="button" value="触发过渡动画" />
<div class="box"></div>
```

在文档中包含一个 box 类的盒子和一个按钮，当单击按钮时，jQuery 脚本都会将盒子的类切换为 change，从而触发过渡动画，演示效果如图 13.24 所示。



默认状态　　JavaScript 事件激活状态

图 13.24　使用 JavaScript 脚本触发动画

上面演示了样式发生变化会导致过渡动画，也可以通过其他方法触发这些更改，包括通过 JavaScript 脚本动态更改。从执行效率来看，事件通常应当通过 JavaScript 触发，简单动画或过渡则应使用 CSS 触发。

13.3　帧　动　画

2012 年 4 月，W3C 发布了 CSS Animations 工作草案，在这个草案中描述了 CSS 关键帧动画的基



本实现方法和属性。目前最新版本的主流浏览器都支持 CSS 帧动画，如 IE 10+、Firefox 和 Opera 均支持不带前缀的动画属性 animation，而旧版浏览器则支持带前缀的动画属性，如 Webkit 引擎支持-webkit-animation 私有属性，Mozilla Gecko 引擎支持-moz-animation 私有属性，Presto 引擎支持-o-animation 私有属性，IE6～IE9 浏览器不支持 animation 属性。

权威参考：http://www.w3.org/TR/css3-animations/。


权威参考


视频讲解

13.3.1　设置关键帧

CSS3 使用@keyframes 定义关键帧。具体用法如下：

```
@keyframes animationname {
    keyframes-selector {
        css-styles;
    }
}
```

参数说明如下。

☑ animationname：定义动画的名称。

☑ keyframes-selector：定义帧的时间位置，也就是动画时长的百分比，合法的值包括 0～100%、from（等价于 0%）、to（等价于 100%）。

☑ css-styles：表示一个或多个合法的 CSS 样式属性。

在动画进行过程中，用户能够多次改变这套 CSS 样式。以百分比或者通过关键词 from 和 to 来定义样式改变发生的时间。为了获得最佳浏览器支持，设计关键帧动画时，应该始终定义 0%和 100%位置帧。最后，为每帧定义动态样式，同时将动画与选择器绑定。

【示例】下面示例演示如何让一个小方盒沿着方形框内壁匀速运动，效果如图 13.25 所示。

```
<style>
#wrap {/*定义运动轨迹包含框*/
    position:relative;          /*定义定位包含框，避免小盒子跑到外面运动*/
    border:solid 1px red;
    width:250px; height:250px;
}
#box {/*定义运动小盒的样式*/
    position:absolute;
    left:0; top:0;
    width: 50px; height: 50px;
    background: #93FB40;
    border-radius: 8px;
    box-shadow: 2px 2px 2px #999;
    /*定义帧动画：名称为 ball，动画时长 5s，动画类型为匀速渐变，动画无限播放*/
    animation: ball 5s linear infinite;
}
/*定义关键帧：共包括 5 帧，分别在总时长 0%、25%、50%、75%、100%的位置*/
/*每帧中设置动画属性为 left 和 top，让它们的值匀速渐变，产生运动动画*/
@keyframes ball {
    0% {left:0;top:0;}
    25% {left:200px;top:0;}
    50% {left:200px;top:200px;}
```





Note

```
        75% {left:0;top:200px;}
        100% {left:0;top:0;}
    }
    </style>
    <div id="wrap">
        <div id="box"></div>
    </div>
```

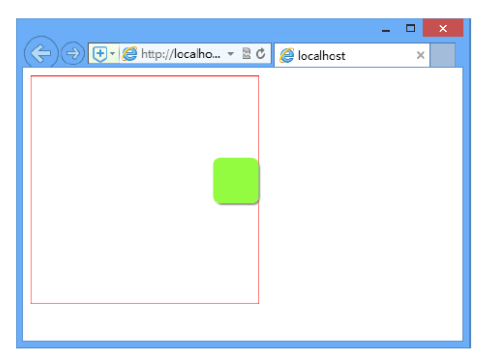


图 13.25　设计小盒子运动动画



视 频 讲 解

13.3.2　设置动画属性

Animations 功能与 Transition 功能相同，都是通过改变元素的属性值来实现动画效果的。它们的区别在于：使用 Transitions 功能时只能通过指定属性的开始值与结束值，然后在这两个属性值之间进行平滑过渡的方式来实现动画效果，因此不能实现比较复杂的动画效果；而 Animations 则通过定义多个关键帧以及定义每个关键帧中元素的属性值来实现更为复杂的动画效果。

1．定义动画名称

使用 animation-name 属性可以定义 CSS 动画的名称，语法如下：

```
animation-name:none | IDENT [, none | IDENT]*;
```

初始值为 none，定义一个适用的动画列表。每个名字用来选择动画关键帧，提供动画的属性值。如果名称是 none，那么就不会有动画。

2．定义动画时间

使用 animation-duration 属性可以定义 CSS 动画播放时间，语法如下：

```
animation-duration:<time> [, <time>]*;
```

在默认情况下该属性值为 0，这意味着动画周期是直接的，即不会有动画。当值为负值时，则被视为 0。

3．定义动画类型

使用 animation-timing-function 属性可以定义 CSS 动画类型，语法如下：

```
animation-timing-function:ease | linear | ease-in | ease-out | ease-in-out | cubicbezier(<number>, <number>, number>, <number>) [, ease | linear |ease-in | ease-out | ease-in-out | cubic-bezier(<number>, <number>,<number>, <number>)]*
```





初始值为 ease，取值说明可参考 13.2.4 节介绍的过渡动画类型。

4．定义延迟时间

使用 animation-delay 属性可以定义 CSS 动画延迟播放的时间，语法如下：

```
animation-delay: <time> [, <time>]*;
```

该属性允许一个动画开始执行一段时间后才被应用。当动画延迟时间为 0，即默认动画延迟时间，则意味着动画将尽快执行，否则该值指定将延迟执行的时间。

5．定义播放次数

使用 animation-iteration-count 属性定义 CSS 动画的播放次数，语法如下：

```
animation-iteration-count: infinite | <number> [, infinite | <number>]*;
```

默认值为 1，这意味着动画将从开始到结束播放一次。infinite 表示无限次，即 CSS 动画永远重复。如果取值为非整数，将导致动画结束一个周期的一部分。如果取值为负值，则将导致在交替周期内反向播放动画。

6．定义播放方向

使用 animation-direction 属性定义 CSS 动画的播放方向，基本语法如下：

```
animation-direction: normal | alternate [, normal | alternate]*;
```

默认值为 normal。当为默认值时，动画的每次循环都向前播放。另一个值是 alternate，设置该值则表示第偶数次向前播放，第奇数次向反方向播放。

7．定义播放状态

使用 animation-play-state 属性定义动画正在运行还是暂停，语法如下：

```
animation-play-state: paused|running;
```

初始值为 running。其中 paused 定义动画已暂停，running 定义动画正在播放。

提示：可以在 JavaScript 中使用 animation-play-state 属性，这样就能在播放过程中暂停动画。在 JavaScript 脚本中用法如下：

```
object.style.animationPlayState="paused"
```

8．定义播放外状态

使用 animation-fill-mode 属性定义动画外状态，语法如下：

```
animation-fill-mode: none | forwards | backwards | both [, none | forwards | backwards | both]*
```

初始值为 none，如果提供多个属性值，以逗号进行分隔。取值说明如下。

☑ none：不设置对象动画之外的状态。
☑ forwards：设置对象状态为动画结束时的状态。
☑ backwards：设置对象状态为动画开始时的状态。
☑ both：设置对象状态为动画结束或开始的状态。

【示例】下面示例设计一个小球，并定义它水平向左运动，动画结束之后，再返回起始点位置，效果如图 13.26 所示。





Note

```
<style>
/*启动运动的小球，并定义动画结束后返回*/
.ball{
      width: 50px; height: 50px;
      background: #93FB40;
      border-radius: 100%;
      box-shadow:2px 2px 2px #999;
      animation:ball 1s ease backwards;
}
/*定义小球水平运动关键帧*/
@keyframes ball{
      0%{transform:translate(0,0);}
      100%{transform:translate(400px);}
}
</style>
<div class="ball"></div>
```

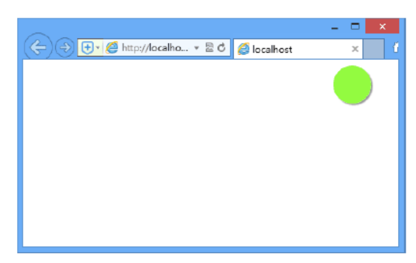


图 13.26　设计运动小球最后返回起始点位置

13.4　案 例 实 战

本节将通过多个案例帮助读者上机练习和提升 CSS3 动画设计技法。



视频讲解

13.4.1　设计图形

【示例 1】设计菱形。制作菱形的方法有很多种，本例结合使用 transform 属性和 rotate()函数，使两个正反三角形上下显示，效果如图 13.27 所示。

```
#shape {
      width: 120px; height: 120px;
      background: #1eff00;
      margin: 60px auto;
      transform: rotate(-45deg);      /*逆时针旋转 45° */
      transform-origin: 0 100%;       /*以右上角为原点进行旋转*/
}
</style>
<div id="shape"></div>
```

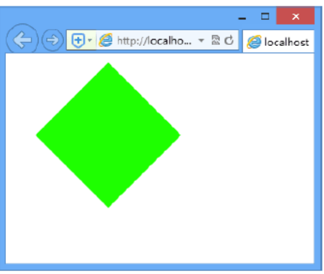
【示例 2】设计平行四边形。使用 transform 属性让长方形倾斜一个角度即可制作平行四边形，效果如图 13.28 所示。





Note

```
<style type="text/css">
#shape {
    width: 200px; height: 120px;
    background: #1eff00;
    margin: 60px auto;
    transform: skew(30deg);
}
</style>
<div id="shape"></div>
```



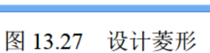


图 13.27　设计菱形

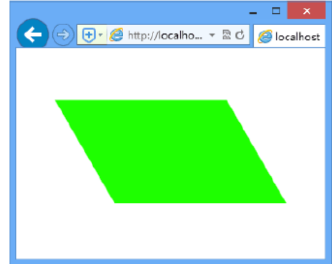
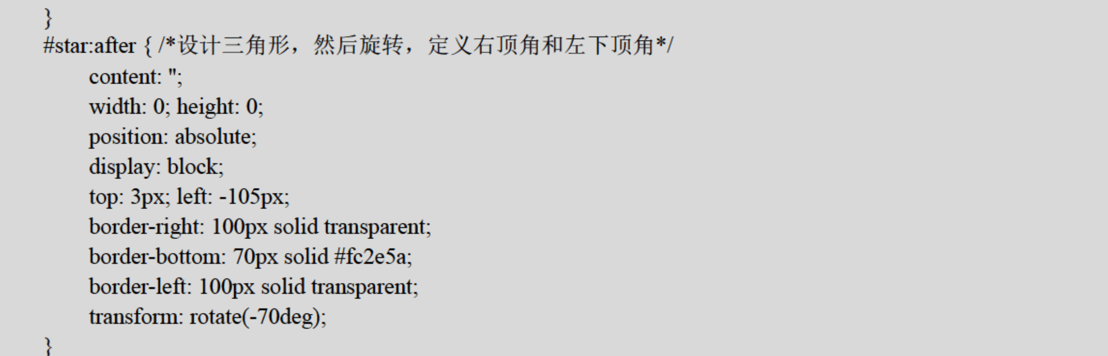


图 13.28　设计平行四边形

【示例 3】设计星形。星形的 HTML 结构也是一个<div id="star">标签，星形的实现方法比较复杂，主要是使用 transform 属性来旋转不同的边，借助:before 和:after 伪对象完成，样式代码如下，效果如图 13.29 所示。

```
<style type="text/css">
#star { /*设计三角形，然后旋转，定义左顶角和右下顶角*/
    width: 0; height: 0;
    margin: 80px auto;
    color: #fc2e5a;
    position: relative;          /*定义定位包含框，后面生成内容根据该框定位*/
    display: block;              /*块显示，避免行内显示出现异常*/
    /*设计三角形*/
    border-right: 100px solid transparent;
    border-bottom: 70px solid #fc2e5a;
    border-left: 100px solid transparent;
    /*旋转三角形*/
    transform: rotate(35deg);
}
#star:before { /*生成三角形，定义向上顶角*/
    content: '';                 /*不包含内容*/
    height: 0; width: 0;
    position: absolute;          /*绝对定位*/
    display: block;              /*块显示，避免行内显示出现异常*/
    top: -45px; left: -65px;     /*固定到顶部位置显示*/
    /*设计三角形*/
    border-bottom: 80px solid #fc2e5a;
    border-left: 30px solid transparent;
    border-right: 30px solid transparent;
    /*旋转三角形*/
    transform: rotate(-35deg);
```





```
}
#star:after { /*设计三角形，然后旋转，定义右顶角和左下顶角*/
      content: '';
      width: 0; height: 0;
      position: absolute;
      display: block;
      top: 3px; left: -105px;
      border-right: 100px solid transparent;
      border-bottom: 70px solid #fc2e5a;
      border-left: 100px solid transparent;
      transform: rotate(-70deg);
}
</style>

<div id="star"></div>
```

【示例 4】下面示例设计一个心形。心形的制作比较复杂，可以使用伪对象，分别将伪对象旋转不同的角度，并修改 transform-origin 属性来设计对象的旋转中心点。示例样式代码如下，效果如图 13.30 所示。

```
<style type="text/css">
#heart {position: relative; margin:50px auto; width:120px;}
#heart:before, #heart:after {
      content: "";
      width: 70px; height: 115px;
      position: absolute;
      background: red;
      left: 70px; top: 0;
      border-radius: 50px 50px 0 0;
      transform: rotate(-45deg);
      transform-origin: 0 100%;
}
#heart:after {
      left: 0;
      transform: rotate(45deg);
      transform-origin: 100% 100%;
}
</style>

<div id="heart"></div>
```

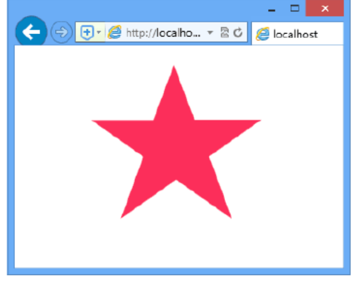


图 13.29　设计星形

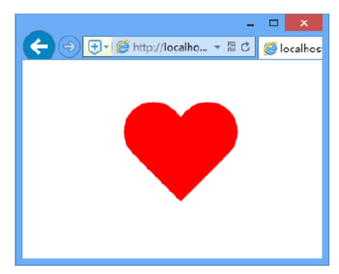


图 13.30　设计心形



提示：配合使用变形函数和伪对象选择器，还可以设计出更多复杂图形。

13.4.2　设计冒泡背景按钮

本例应用 CSS3 过渡动画特效，为按钮背景图像定义动态移动效果，设计当鼠标经过时，按钮背景绚丽多彩，不断产生冒泡动画效果，如图 13.31 所示。

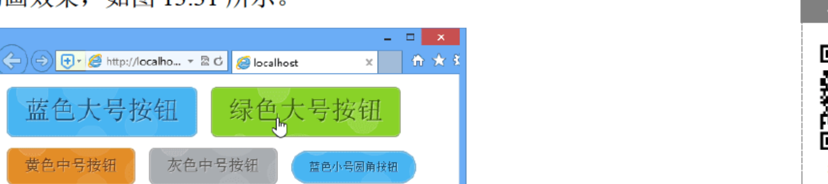


图 13.31　设计背景冒泡效果的按钮样式

【操作步骤】

第 1 步，设计按钮基本样式。

```
.button{
    font:15px Calibri, Arial, sans-serif;
    /*半透明的文本阴影*/
    text-shadow:1px 1px 0 rgba(255,255,255,0.4);
    /*重写默认下画线的链接样式*/
    text-decoration:none !important;
    white-space:nowrap;          /*禁止文本换行显示*/
    display:inline-block;        /*行内块显示*/
    vertical-align:baseline;     /*垂直基线对齐*/
    position:relative;           /*相对定位*/
    cursor:pointer;              /*鼠标指针为手形*/
    padding:10px 20px;           /*增加按钮内空间*/
    background-repeat:no-repeat;
    /*下面两个规则是回退，以防浏览器不支持多重背景*/
    background-position:bottom left;
    background-image:url('images/button_bg.png');
    /*多重背景。背景图像在颜色类中单独定义*/
    background-position:bottom left, top right, 0 0, 0 0;
    background-clip:border-box;
    /*设计圆角*/
    border-radius:8px;
    /*添加 1px 的高亮效果*/
    box-shadow:0 0 1px #fff inset;
    /*设计 CSS 过渡动画，动画属性为背景图像的位置*/
    transition:background-position 1s;
}
```

第 2 步，设计鼠标经过时的动态样式。

```
.button:hover {
    background-position: top left; /*回退技术，兼容浏览器不支持多背景*/
```

Note

视频讲解



```
        background-position: top left, bottom right, 0 0, 0 0;
    }
```

第 3 步，设计激活时，按钮下沉样式。



Note

```
.button:active {bottom: -1px;}
```

第 4 步，设计按钮大小号类样式。

```
.button.button_big {font-size: 30px;}
.button.button_medium {font-size: 18px;}
.button.button_small {font-size: 13px;}
```

第 5 步，设计圆角按钮类样式。

```
.button.button_rounded {border-radius: 4em;}
```

第 6 步，设计按钮主题类样式。

```
.button_blue.button {
    color: #0f4b6d !important;
    border: 1px solid #84acc3 !important;
    /*回退背景色颜色*/
    background-color: #48b5f2;
    /*定义多背景图*/
    background-image: url('images/button_bg.png'), url('images/button_bg.png'), radial-gradient(center bottom, circle, rgba(89,208,244,1) 0, rgba(89,208,244,0) 100px), linear-gradient(#4fbbf7, #3faeeb);
}
.button_blue.button:hover { /*定义鼠标经过时多背景图样式*/
    background-color: #63c7fe;
    background-image: url('images/button_bg.png'), url('images/button_bg.png'), radial-gradient(center bottom, circle, rgba(109,217,250,1) 0, rgba(109,217,250,0) 100px), linear-gradient(#63c7fe, #58bef7);
}
```



视 频 讲 解

13.4.3 设计动画效果菜单

本例利用 CSS3 过渡动画设计一个界面切换的导航菜单，当鼠标经过菜单项时，会以动画形式从中文界面缓慢翻转到英文界面，或者从英文界面翻转到中文界面，效果如图 13.32 所示。

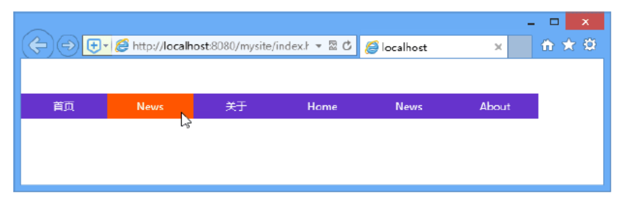


图 13.32　设计动画翻转菜单样式

【操作步骤】

第 1 步，设计菜单结构。在每个菜单项（<div class="menu1">）中包含两个子标签：<div class= "one">和<div class="two">，设计菜单项仅显示一个子标签，当鼠标经过时，翻转显示另一个子标签。

```
<div>
    <div class="menu1">
```



```
        <div class="one"><a href="#">首页</a></div>
        <div class="two"><a href="#">Home</a></div>
    </div>
    <div class="menu1">
        <div class="one"><a href="#">新闻</a></div>
        <div class="two"><a href="#">News</a></div>
    </div>
    <div class="menu1">
        <div class="one"><a href="#">关于</a></div>
        <div class="two"><a href="#">About</a></div>
    </div>
</div>
```



第 2 步，设计菜单项的样式。固定大小，相对定位，禁止内容溢出容器，向左浮动，定义并列显示。

```
.menu1 {
    width: 100px; height: 30px;
    position: relative;
    font-family: 微软雅黑; font-size: 12px; color: #fff;
    overflow: hidden;
    float: left;
}
```

第 3 步，设计每个菜单项中子标签<div class="one">和 <div class="two">的样式。定义它们与菜单项相同大小，这样就只能够显示一个子标签；为了方便控制，定义它们为绝对定位，包含文本水平居中和垂直居中；最后定义过渡动画时间为 0.3s，加速到减速显示。

```
.menu1 div {
    width: 100px; height: 30px;
    line-height: 30px;   text-align: center;
    position: absolute;
    transition: all 0.3s ease-in-out;
}
```

第 4 步，设计过渡动画样式。本例设计过渡演示属性为 left、top 和 bottom，当鼠标经过时，改变定位属性的值，实现菜单项动态翻转效果。

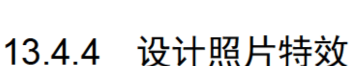
```
.menu1 .one {
    top: 0; left: 0;
    z-index: 1;
    background: #63C; color: #FFF;
}
.menu1:hover .one {top: -30px; left: 0;}
.menu1 .two {
    bottom: -30px; left: 0;
    z-index: 2;
    background: #f50; color: #FFF;
}
.menu1:hover .two {bottom: 0px; left: 0;}
```



视 频 讲 解



Note



13.4.4 设计照片特效

本例使用 CSS3 阴影、透明效果，以及变换，设计图片随意贴在墙上效果，当鼠标移动到图片上时，图片会自动放大并垂直排列，演示效果如图 13.33 所示。

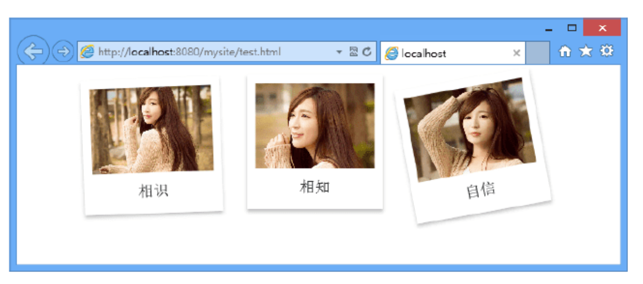


图 13.33 设计挂图效果

案例主要代码如下：

```
<style type="text/css">
ul.polaroids li {display: inline;}
ul.polaroids a {
      display: inline; float: left;
      margin: 0 0 50px 60px; padding: 12px;
      text-align: center;
      text-decoration: none; color: #333;
       /*为图片外框设计阴影效果*/
      box-shadow: 0 3px 6px rgba(0, 0, 0, .25);
       /*设置过渡动画：过渡属性为 transform，时长为 0.15s，线性渐变*/
      transition: -webkit-transform .15s linear;
      /*顺时针旋转 2° */
      transform: rotate(-2deg);
}
ul.polaroids img { /*统一图片基本样式*/
      display: block;
      height: 100px;
      border: none;
      margin-bottom: 12px;
}
/*利用图片的 tittle 属性，添加图片显示标题*/
ul.polaroids a:after {content: attr(title);}
/*偶数图片倾斜显示*/
ul.polaroids li:nth-child(even) a {
      transform: rotate(10deg);       /*逆时针旋转 10° */
}
ul.polaroids li a:hover {
       /*放大对象 1.25 倍*/
      transform: scale(1.25);
      box-shadow: 0 3px 6px rgba(0, 0, 0, .5);
}
</style>
```



```
<ul class="polaroids">
    <li> <a href="1" title="相识"> <img src="images/1.jpg" alt="相识"> </a> </li>
    <li> <a href="2" title="相知"> <img src="images/2.jpg" alt="相知"> </a> </li>
    <li> <a href="3" title="自信"> <img src="images/3.jpg" alt="自信"> </a> </li>
</ul>
```







视频讲解

13.4.5 设计立体盒子

【示例 1】下面示例使用 2D 多重变换制作一个正方体，演示效果如图 13.34 所示。

```
<style type="text/css">
body{padding:20px 0 0 100px;}
.side {
    height: 100px; width: 100px;
    position: absolute;
    font-size: 20px; font-weight: bold; line-height: 100px; text-align: center; color: #fff;
    text-shadow: 0 -1px 0 rgba(0,0,0,0.2);
    text-transform: uppercase;
}
.top {/*顶面*/
    background: red;
    transform: rotate(-45deg) skew(15deg, 15deg);
}
.left {/*左侧面*/
    background: blue;
    transform: rotate(15deg) skew(15deg, 15deg) translate(-50%, 100%);
}
.right {/*右侧面*/
    background: green;
    transform: rotate(-15deg) skew(-15deg, -15deg) translate(50%, 100%);
}
</style>
<div class="side top">顶面</div>
<div class="side left">左侧面</div>
<div class="side right">右侧面</div>
```

【示例 2】下面示例使用 3D 多重变换制作一个正方体，演示效果如图 13.35 所示。

```
<style type="text/css">
.stage {/*定义画布样式*/
    width: 300px; height: 300px; margin: 100px auto; position: relative;
    perspective: 300px;
}
/*定义盒子包含框样式*/
.container {transform-style: preserve-3d;}
/*定义盒子六面基本样式*/
.side {
    background: rgba(255,0,0,0.3);
    border: 1px solid red;
    font-size: 60px; font-weight: bold; color: #fff; text-align: center;
    height: 196px; line-height: 196px; width: 196px;
```





Note

```
        position: absolute;
        text-shadow: 0 -1px 0 rgba(0,0,0,0.2);
        text-transform: uppercase;
    }
    .front {/*使用 3D 变换制作前面*/
        transform: translateZ(100px);
    }
    .back {/*使用 3D 变换制作后面*/
        transform: rotateX(180deg) translateZ(100px);
    }
    .left {/*使用 3D 变换制作左面*/
        transform: rotateY(-90deg) translateZ(100px);
    }
    .right {/*使用 3D 变换制作右面*/
        transform: rotateY(90deg) translateZ(100px);
    }
    .top {/*使用 3D 变换制作顶面*/
        transform: rotateX(90deg) translateZ(100px);
    }
    .bottom {/*使用 3D 变换制作底面*/
        transform: rotateX(-90deg) translateZ(100px);
    }
    </style>
    <div class="stage">
        <div class="container">
            <div class="side front">前面</div>
            <div class="side back">背面</div>
            <div class="side left">左面</div>
            <div class="side right">右面</div>
            <div class="side top">顶面</div>
            <div class="side bottom">底面</div>
        </div>
    </div>
```

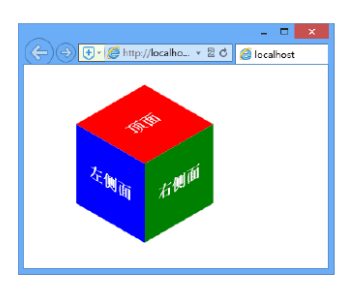


图 13.34　设计 2D 变换盒子

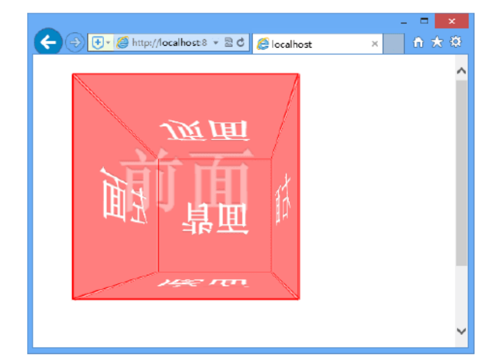


图 13.35　设计 3D 盒子



视频讲解

13.4.6　设计旋转盒子

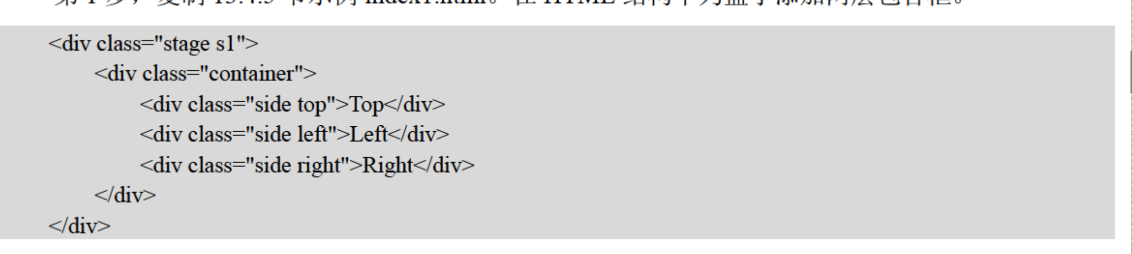
以 13.4.5 节示例为基础，使用 animation 属性设计盒子旋转显示。



【示例 1】本例使用 2D 制作一个正方体，然后设计它在鼠标经过时沿 Y 轴旋转，演示效果如图 13.36 所示。

【操作步骤】

第 1 步，复制 13.4.5 节示例 index1.html。在 HTML 结构中为盒子添加两层包含框。



```
<div class="stage s1">
    <div class="container">
        <div class="side top">Top</div>
        <div class="side left">Left</div>
        <div class="side right">Right</div>
    </div>
</div>
```

第 2 步，在内部样式表中定义关键帧。

```
/*定义关键帧动画*/
@keyframes spin{/*标准模式*/
    0%{transform:rotateY(0deg)}
    100%{transform:rotateY(360deg)}
}
```

第 3 步，设计 3D 变换的透视距离和变换类型，即启动 3D 变换。

```
/*定义盒子所在画布框的样式*/
.stage {perspective: 1200px;}
/*定义盒子包含框样式*/
.container {transform-style: preserve-3d;}
```

第 4 步，定义动画触发方式。

```
/*定义鼠标经过盒子时触发线性变形动画，动画时间为 5s，持续播放*/
.container:hover{animation:spin 5s linear infinite;}
```

本例完整代码请参考本书源代码。

【示例 2】本例使用 3D 制作一个正方体，然后设计它在鼠标经过时沿 Y 轴旋转，演示效果如图 13.37 所示。

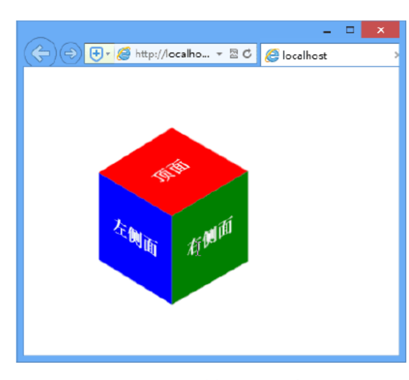


图 13.36　设计旋转的 3D 盒子（1）

图 13.37　设计旋转的 3D 盒子（2）

【操作步骤】

第 1 步，在内部样式表中定义关键帧。



```
/*定义关键帧动画*/
@keyframes spin {
    0% {transform:rotateY(0deg)}
    100% {transform:rotateY(360deg)}
}
```



Note

第 2 步，设计 3D 变换的透视距离和变换类型，即启动 3D 变换。

```
/*定义画布样式*/
.stage {perspective: 300px;}
/*定义盒子包含框样式*/
.container {transform-style: preserve-3d;}
```

第 3 步，定义动画触发方式。

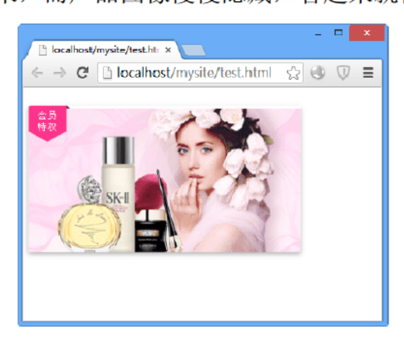
```
/*定义鼠标经过时触发盒子旋转动画*/
.container:hover {animation: spin 5s linear infinite;}
```

本例完整代码请参考本书源代码。

视 频 讲 解

13.4.7　设计翻转广告

本例设计当鼠标移动到产品图片上时，产品信息翻转滑出，效果如图 13.38 所示。在默认状态下只显示产品图片，而产品信息不可见。当鼠标移动到产品图像上时，产品图像慢慢往上旋转，使产品信息展示出来，而产品图像慢慢隐藏，看起来就像是一个旋转的盒子。



默认状态



翻转状态

图 13.38　设计 3D 翻转广告

案例主要代码如下：

```
<style type="text/css">
/*定义包含框样式*/
.wrapper {
    display: inline-block; width: 345px; height: 186px; margin: 1em auto; cursor: pointer; position: relative;
    /*定义 3D 元素距视图的距离*/
    perspective: 4000px;
}
/*定义旋转元素样式：3D 动画，动画时间为 0.6s*/
.item {
    height: 186px;
    transform-style: preserve-3d;
```






```
        transition: transform .6s;
}
*定义鼠标经过时触发动画，并定义旋转形式*/
.item:hover {
        transform: translateZ(-50px) rotateX(95deg);
}
.item:hover img {box-shadow: none; border-radius: 15px;}
.item:hover .information {box-shadow: 0px 3px 8px rgba(0,0,0,0.3); border-radius: 15px;}
/*定义广告图的动画形式和样式*/
.item>img {
        display: block; position: absolute; top: 0; border-radius: 3px;box-shadow: 0px 3px 8px rgba(0,0,0,0.3);
        transform: translateZ(50px);
        transition: all .6s;
}
/*定义广告文字的动画形式和样式*/
.item .information {
        position: absolute; top: 0; height: 186px; width: 345px; border-radius: 15px;
        transform: rotateX(-90deg) translateZ(50px);
        transition: all .6s;
}
</style>

<div class="wrapper">
        <div class="item">
                <img src="images/1.png" />
                <span class="information"><img src="images/2.png" /></span>
        </div>
</div>
```

13.4.8　设计跑步效果



本例设计一个跑步动画效果，主要使用 CSS3 帧动画控制一张序列人物跑步的背景图像，在页面固定“镜头”中快速切换实现动画效果，如图 13.39 所示。

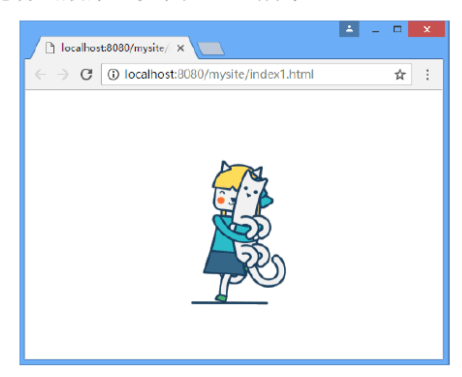


图 13.39　设计跑步效果

【操作步骤】

第 1 步，设计舞台场景结构。新建 HTML 文档，保存为 index1.html。输入下面代码：





```
<div class="charector-wrap " id="js_wrap">
    <div class="charector"></div>
</div>
```

第 2 步，设计舞台基本样式。其中导入的图片是一个序列跑步人物集合，如图 13.40 所示。

```
.charector-wrap {
    position: relative;
    width: 180px;
    height: 300px;
    left: 50%;
    margin-left: -90px;
}
.charector{
    position: absolute;
    width: 180px;
    height:300px;
    background: url(img/charector.png) 0 0 no-repeat;
}
```



图 13.40　小人序列集合

本例主要设计任务就是让序列小人仅显示一个，然后通过 CSS3 动画，让他们快速闪现在指定限定框中。

第 3 步，设计动画关键帧。

```
@keyframes person-normal{/*跑步动画名称*/
    0% {background-position: 0 0;}
    14.3% {background-position: -180px 0;}
    28.6% {background-position: -360px 0;}
    42.9% {background-position: -540px 0;}
    57.2% {background-position: -720px 0;}
    71.5% {background-position: -900px 0;}
    85.8% {background-position: -1080px 0;}
    100% {background-position: 0 0;}
}
```

第 4 步，设置动画属性。

```
.charector{
    animation-iteration-count: infinite;/*动画无限播放*/
    animation-timing-function:step-start;/*马上跳到动画每一结束帧的状态*/
}
```

第 5 步，启动动画，并设置动画频率。

```
/*启动动画，并控制跑步动作频率*/
.charector{
```



```
        animation-name: person-normal;
        animation-duration: 800ms;
    }
```

13.4.9 设计折叠面板



Note



视频讲解

本案例使用 CSS3 的目标伪类（:target）设计折叠面板效果，没有使用 JavaScript 脚本，使用过渡属性设计滑动效果。折叠动画效果如图 13.41 所示。

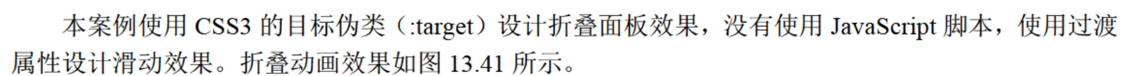




图 13.41　设计折叠面板

案例主要代码如下：

```
<style type="text/css">
/*定义折叠框外框样式*/
.accordion {
        background: #eee;
        border: 1px solid #999;
        margin: 2em;}
/*定义折叠框标题栏样式*/
.accordion h2 {
        margin: 0;
        padding: 12px 0;
        background:#CCC}
/*定义折叠框内容框样式*/
.accordion .section {
        border-bottom: 1px solid #ccc;
        background: #fff;}
/*定义折叠框选项标题栏样式*/
.accordion h3 {
        margin:0;
        padding:0;
        background: #eee;
        padding:3px 1em;}
/*定义折叠框选项标题栏超链接样式*/
.accordion h3 a {
```





Note

```
        font-weight: normal;
        text-decoration:none;}
/*当获得目标焦点时，粗体显示选项标题栏文字*/
.accordion :target h3 a {font-weight: bold;}
/*选项栏标题对应的选项子框样式*/
.accordion h3 + div {
        height: 0;
        padding:0 1em;
        overflow: hidden;
        /*定义过渡对象为高度，过渡时间为 0.3s，渐显显示*/
        transition: height 0.3s ease-in;}
.accordion h3 + div img {margin:4px;}
/*当获得目标焦点时，子选项内容框样式*/
.accordion :target h3 + div {
        /*当获取目标之后，高度为 300px*/
        height:300px;
        overflow:auto;}
</style>
<div class="accordion">
        <h2>我爱买</h2>
        <div id="one" class="section">
                <h3> <a href="#one">爱逛</a> </h3>
                <div><img src="images/11.png"></div>
        </div>
        <div id="two" class="section">
                <h3> <a href="#two">爱美丽</a> </h3>
                <div><img src="images/22.png"></div>
        </div>
        <div id="three" class="section">
                <h3> <a href="#three">爱吃</a> </h3>
                <div><img src="images/33.png"></div>
        </div>
</div>
```

13.5 在线练习

本节练习 CSS3 动画一般设计方法，培养灵活应用交互式动态样式的基本能力。感兴趣的读者可以扫码练习。

在线练习



使用CSS3媒体查询

2017年9月，W3C发布了媒体查询（Media Query Level 4）候选推荐标准规范，它扩展了已经发布的媒体查询的功能。该规范用于CSS的@media规则，可以为文档设定特定条件的样式，也可用于HTML、JavaScript等语言中。

权威参考：http://www.w3.org/TR/css3-mediaqueries/。

权威参考

【学习重点】

- 了解CSS3媒体类型。
- 正常使用媒体查询的条件规则。
- 设计响应不同设备的网页布局。



14.1 媒体查询基础


Note

媒体查询可以根据设备特性，如屏幕宽度、高度和设备方向（横向或纵向），为设备定义独立的CSS 样式表。一个媒体查询由一个可选的媒体类型和零个或多个限制范围的表达式组成，如宽度、高度和颜色。

14.1.1 媒体类型和媒体查询

CSS2 提出媒体类型（Media Type）的概念，它允许为样式表设置限制范围的媒体类型。例如，仅供打印的样式表文件、仅供手机渲染的样式表文件、仅供电视渲染的样式表文件等，具体说明如表 14.1 所示。

表 14.1 CSS 媒体类型

类　型	支持的浏览器	说　明
aural	Opera	用于语音和音乐合成器
braille	Opera	用于触觉反馈设备
handheld	Chrome，Safari，Opera	用于小型或手持设备
print	所有浏览器	用于打印机
projection	Opera	用于投影图像，如幻灯片
screen	所有浏览器	用于屏幕显示器
tty	Opera	用于使用固定间距字符格的设备，如电传打字机和终端
tv	Opera	用于电视类设备
embossed	Opera	用于凸点字符（盲文）印刷设备
speech	Opera	用于语音类型
all	所有浏览器	用于所有媒体设备类型

通过 HTML 标签属性 media 定义样式表的媒体类型，具体方法如下：

☑ 定义外部样式表文件的媒体类型。

```
<link href="csss.css" rel="stylesheet" type="text/css" media="handheld" />
```

☑ 定义内部样式表文件的媒体类型。

```
<style type="text/css" media="screen">
...
</style>
```

CSS3 在媒体类型基础上，提出了 Media Queries（媒体查询）的概念。媒体查询比 CSS2 的媒体类型功能更强大、更加完善。两者主要区别：媒体查询是一个值或一个范围的值，而媒体类型仅仅是设备的匹配。媒体类型可以帮助用户获取以下数据：

☑ 浏览器窗口的宽和高。
☑ 设备的宽和高。
☑ 设备的手持方向：横向还是竖向。
☑ 分辨率。



例如，下面这条导入外部样式表的语句，在 media 属性中设置媒体查询的条件(max-width: 600px)：当屏幕宽度小于或等于 600px 时，则调用 small.css 样式表来渲染页面。

```
<link rel="stylesheet" media="screen and (max-width: 600px)" href="small.css" />
```



14.1.2　@media

CSS3 使用@media 规则定义媒体查询，简化语法格式如下：

```
@media [only | not]? <media_type> [and <expression>]* | <expression> [and <expression>]*{
    /*CSS 样式列表*/
}
```

参数简单说明如下。

☑ <media_type>：指定媒体类型，具体说明参考表 14.1。

☑ <expression>：指定媒体特性。放在一对圆括号中，如(min-width:400px)。

☑ 逻辑运算符，如 and（逻辑与）、not（逻辑否）、only（兼容设备）等。

媒体特性包括 13 种，接受单个的逻辑表达式作为值或者没有值。大部分特性接受 min 或 max 的前缀，用来表示大于等于或者小于等于的逻辑，以此避免使用大于号（>）和小于号（<）字符。下面是各种媒体特性的简单说明。

1．颜色（color）

指定输出设备每个像素单元的比特值。接受 min/max 前缀。例如，向每个颜色单元至少有 4 个比特的设备应用样式表：

```
@media all and (min-color: 4) {...}
```

2．颜色索引（color-index）

指定输出设备中颜色查询表中的条目数量。接受 min/max 前缀。例如，向所有使用索引颜色的设备应用样式表：

```
@media all and (color-index) {...}
```

3．宽高比（aspect-ratio）

指定输出设备目标显示区域的宽高比。取值包含两个以“/”分隔的正整数，代表水平像素数与垂直像素数的比例。接受 min/max 前缀。例如，下面为显示区域宽高比至少为一比一的设备选择了一个特殊的样式表：

```
@media screen and (min-aspect-ratio: 1/1) {...}
```

4．设备宽高比（device-aspect-ratio）

指定输出设备的宽高比。接受 min/max 前缀。例如，下面为宽屏设备选择了一个特殊的样式表：

```
@media screen and (device-aspect-ratio: 16/9), screen and (device-aspect-ratio: 16/10) {...}
```

5．设备高度（device-height）

指定输出设备的高度。接受 min/max 前缀。例如，在最大宽度“800px”的屏幕上应用样式表：

```
<link rel="stylesheet" media="screen and (max-device-width: 799px)" />
```





6．设备宽度（device-width）

指定输出设备的宽度。接受 min/max 前缀。

7．网格（grid）

判断输出设备是网格设备还是位图设备。如果设备是基于网格的（如电传打字机），该值为 1，否则为 0。例如，向一个 15 字符宽度或更窄的手持设备应用样式：

```
@media handheld and (grid) and (max-width: 15em) {...}
```

8．高度（height）

指定输出设备渲染区域的高度。接受 min/max 前缀。

9．黑白（monochrome）

指定一个黑白设备每个像素的比特数。接受 min/max 前缀。例如，向所有黑白设备应用样式表：

```
@media all and (monochrome) {...}
```

10．方向（orientation）

指定设备处于横屏（宽度大于高度）模式还是竖屏（高度大于宽度）模式。取值包括 landscape（横屏）和 portrait（竖屏）。例如，向竖屏设备应用样式表：

```
@media all and (orientation: portrait) {...}
```

11．分辨率（resolution）

指定输出设备的分辨率。接受 min/max 前缀。例如，为每英寸至少 300 点的打印机应用样式：

```
@media print and (min-resolution: 300dpi) {...}
```

12．扫描（scan）

指定电视输出设备的扫描过程。取值包括 progressive 和 interlace（交错）。例如，向以顺序方式扫描的电视机上应用样式表：

```
@media tv and (scan: progressive) {...}
```

13．宽度（width）

指定输出设备渲染区域的宽度。接受 min/max 前缀。例如，向宽度在 500px 和 800px 之间的屏幕应用样式表：

```
@media screen and (min-width: 500px) and (max-width: 800px) {...}
```

在 CSS 样式的开头必须定义@media 关键字，然后指定媒体类型，再指定媒体特性。媒体特性的格式与样式的格式相似，分为两部分，由冒号分隔，冒号前指定媒体特性，冒号后指定该特性的值。

例如，下面语句指定了当设备显示屏幕宽度小于 640px 时所使用的样式。

```
@media screen and (max-width: 639px) {
    /*样式代码*/
}
```

可以使用多个媒体查询将同一个样式应用于不同的媒体类型和媒体特性中，媒体查询之间通过逗





号分隔，类似于选择器分组。

```
@media handheld and (min-width:360px),screen and (min-width:480px) {
    /*样式代码*/
}
```

可以在表达式中加上 not、only 和 and 等逻辑运算符。

```
//下面样式代码将被使用在除便携设备之外的其他设备或非彩色便携设备中
@media not handheld and (color) {
    /*样式代码*/
}
//下面样式代码将被使用在所有非彩色设备中
@media all and (not color) {
    /*样式代码*/
}
```

only 运算符能够让不支持媒体查询但支持媒体类型的设备，忽略表达式中的样式。例如：

```
@media only screen and (color) {
    /*样式代码*/
}
```

对于支持媒体查询的设备来说，能够正确地读取其中的样式，仿佛 only 运算符不存在一样；对于不支持媒体查询但支持媒体类型的设备（如 IE8）来说，可以识别@media screen 关键字，但是由于先读取的是 only 运算符，而不是 screen 关键字，将忽略这个样式。

提示：媒体查询也可以用在@import 规则和<link>标签中。例如：

```
@import url(example.css) screen and (width:800px);
/*下面代码定义了如果页面通过屏幕呈现，且屏幕宽度不超过 480px，则加载 shetland.css 样式表*/
<link rel="stylesheet" type="text/css" media="screen and (max-device-width: 480px)" href="shetland.css" />
```

14.1.3　应用@media

【示例 1】and 运算符用于符号两边规则均满足条件的匹配。

```
@media screen and (max-width: 600px) {
    /*匹配宽度小于等于 600px 的屏幕设备*/
}
```

【示例 2】not 运算符用于取非，所有不满足该规则的均匹配。

```
@media not print {
    /*匹配除了打印机以外的所有设备*/
}
```

注意：not 仅应用于整个媒体查询。

```
@media not all and (max-width: 500px) {}
/*等价于*/
@media not (all and (max-width: 500px)) {}
/*而不是*/
@media (not all) and (max-width: 500px) {}
```



在逗号媒体查询列表中，not 仅会否定它所在的媒体查询，而不影响其他的媒体查询。

如果在复杂的条件中使用 not 运算符，要显式添加小括号，避免歧义。



Note

【示例 3】逗号（,）相当于 or 运算符，用于两边有一条规则满足则匹配。

```
@media screen, (min-width: 800px) {
    /*匹配屏幕或者宽度大于等于 800px 的设备*/
}
```

【示例 4】在媒体类型中，all 是默认值，匹配所有设备。

```
@media all {
    /*可以过滤不支持 media 的浏览器*/
}
```

常用的媒体类型还有 screen 匹配屏幕显示器、print 匹配打印输出，更多媒体类型可以参考表 14.1。

【示例 5】使用媒体查询时，必须要加括号，一个括号就是一个查询。

```
@media (max-width: 600px) {
    /*匹配界面宽度小于等于 600px 的设备*/
}
@media (min-width: 400px) {
    /*匹配界面宽度大于等于 400px 的设备*/
}
@media (max-device-width: 800px) {
    /*匹配设备（不是界面）宽度小于等于 800px 的设备*/
}
@media (min-device-width: 600px) {
    /*匹配设备（不是界面）宽度大于等于 600px 的设备*/
}
```

提示：在设计手机网页时，应该使用 device-width/device-height，因为手机浏览器默认会对页面进行一些缩放，如果按照设备宽高来进行匹配，会更接近预期的效果。

【示例 6】媒体查询允许相互嵌套，这样可以优化代码，避免冗余。

```
@media not print {
    /*通用样式*/
    @media (max-width: 600px) {
        /*此条匹配宽度小于等于 600px 的非打印机设备*/
    }
    @media (min-width: 600px) {
        /*此条匹配宽度大于等于 600px 的非打印机设备*/
    }
}
```

【示例 7】在设计响应式页面时，用户应该根据实际需要，先确定自适应分辨率的阈值，也就是页面响应的临界点。

```
@media (min-width: 768px){
    /*>=768px 的设备*/
}
@media (min-width: 992px){
    /*>=992px 的设备*/
```



```
}
@media (min-width: 1200){
    /*>=1200px 的设备*/
}
```

注意：下面样式顺序是错误的，因为后面的查询范围将覆盖前面的查询范围，导致前面的媒体查询失效。

```
@media (min-width: 1200){ }
@media (min-width: 992px){ }
@media (min-width: 768px){   }
```

因此，当使用 min-width 媒体特性时，应该按从小到大的顺序设计各个阈值。同理，如果使用 max-width，就应该按从大到小的顺序设计各个阈值。

```
@media (max-width: 1199){
    /*<=1199px 的设备*/
}
@media (max-width: 991px){
    /*<=991px 的设备*/
}
@media (max-width: 767px){
    /*<=768px 的设备*/
}
```

【示例 8】用户可以创建多个样式表，以适应不同媒体类型的宽度范围。当然，更有效率的方法是将多个媒体查询整合在一个样式表文件中，这样可以减少请求的数量。

```
@media only screen   and (min-device-width: 320px)   and (max-device-width: 480px) {
    /*样式列表*/
}
@media only screen   and (min-width: 321px) {
    /*样式列表*/
}
@media only screen   and (max-width: 320px) {
    /*样式列表*/
}
```

【示例 9】如果从资源的组织和维护的角度考虑，可以选择使用多个样式表的方式来实现媒体查询，这样做更高效。

```
<link rel="stylesheet" media="screen and (max-width: 600px)" href="small.css" />
<link rel="stylesheet" media="screen and (min-width: 600px)" href="large.css" />
<link rel="stylesheet" media="print" href="print.css" />
```

【示例 10】使用 orientation 属性可以判断设备屏幕当前是横屏（值为 landscape），还是竖屏（值为 portrait）。

```
@media screen and (orientation: landscape) {
    .iPadLandscape {
        width: 30%;
        float: right;
    }
```



```
}
@media screen and (orientation: portrait) {
    .iPadPortrait {clear: both;}
}
```



不过 orientation 属性只在 iPad 上有效，对于其他可转屏的设备（如 iPhone），可以使用 min-device-width 和 max-device-width 来变通实现。

【扩展】

媒体查询仅是一种纯 CSS 方式实现响应式 Web 设计的方法，用户还可以使用 JavaScript 库来实现同样的设计。例如，下载 css3-mediaqueries.js（http://code.google.com/p/css3-mediaqueries-js/），然后在页面中调用。对于老式浏览器（如 IE6～IE8）可以考虑使用 css3-mediaqueries.js 进行兼容。

```
<!--[if lt IE 9]>
<script src="http://css3-mediaqueries-js.googlecode.com/svn/trunk/css3-mediaqueries.js"></script>
<![endif]-->
```

【示例 11】下面代码演示了如何使用 jQuery 来检测浏览器宽度，并为不同的视口调用不同的样式表。

```
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
    $(window).bind("resize", resizeWindow);
    function resizeWindow(e){
        var newWindowWidth = $(window).width();
        if(newWindowWidth < 600){
            $("link[rel=stylesheet]").attr({href: "mobile.css"});
        }
        else if(newWindowWidth > 600){
            $("link[rel=stylesheet]").attr({href: "style.css"});
        }
    }
});
</script>
```

14.2 案例实战

本节将通过几个案例练习 CSS3 媒体查询的网页应用。


视频讲解

14.2.1 判断显示屏幕宽度

本例演示如何正确使用@media 规则，判断当前视口宽度的范围。代码如下：

```
<style type="text/css">
.wrapper {/*定义测试条的样式*/
    padding: 5px 10px; margin: 40px;
    text-align:center; color:#999;
```



```
        border: solid 1px #999;
    }
    .viewing-area span {/*默认情况下隐藏提示文本信息*/
        color: #666;
        display: none;
    }
    /*应用于移动设备，且设备最大宽度为 480px*/
    @media screen and (max-device-width: 480px) {
        .a {background: #ccc;}
    }
    /*显示屏幕宽度小于等于 600px*/
    @media screen and (max-width: 600px) {
        .b {
            background: red; color:#fff;
            border: solid 1px #000;
        }
        span.lt600 {display: inline-block;}
    }
    /*显示屏幕宽度介于 600px 和 900px 之间*/
    @media screen and (min-width: 600px) and (max-width: 900px) {
        .c {
            background: red; color:#fff;
            border: solid 1px #000;
        }
        span.bt600-900 {display: inline-block;}
    }
    /*显示屏幕宽度大于等于 900px*/
    @media screen and (min-width: 900px) {
        .d {
            background: red;   color:#fff;
            border: solid 1px #000;
        }
        span.gt900 {display: inline-block;}
    }
    </style>
    <div class="wrapper a">设备最大宽度为 480 像素。</div>
    <div class="wrapper b">显示屏幕宽度小于等于 600 像素</div>
    <div class="wrapper c">显示屏幕宽度介于 600 像素到 900 像素之间</div>
    <div class="wrapper d">显示屏幕宽度大于等于 900 像素</div>
    <p class="viewing-area">
        <strong>当前显示屏幕宽度：</strong>
        <span class="lt600">小于等于 600px</span>
        <span class="bt600-900">介于 600px - 900px 之间</span>
        <span class="gt900">大于等于 900px</span>
    </p>
```



Note

本例设计当显示屏幕宽度小于等于 600px 时，则高亮显示<div class="wrapper b">测试条，并在底部显示提示信息：小于等于 600px；当显示屏幕宽度介于 600px 和 900px 之间时，则高亮显示<div class="wrapper c">测试条，并在底部显示提示信息：介于 600px～900px 之间；显示屏幕宽度大于等

于 900px 时，则高亮显示<div class="wrapper d">测试条，并在底部显示提示信息：大于等于 900px；当设备宽度小于等于 480px 时，则高亮显示<div class="wrapper a">测试条。演示效果如图 14.1 所示。





显示屏幕宽度小于等于 600px



显示屏幕宽度介于 600px 和 900px 之间



显示屏幕宽度大于等于 900px

图 14.1　使用@media 规则


视 频 讲 解

14.2.2　设计响应式版式

本例在页面中设计 3 个栏目。

☑　<div id="main">：主要内容栏目。

☑　<div id="sub">：次要内容栏目。

☑　<div id="sidebar">：侧边栏栏目。

构建的页面结构如下：

```
<div id="container">
    <div id="wrapper">
        <div id="main">
            <h1>水调歌头・明月几时有</h1>
            <h2>苏轼</h2>
            <p>…</p>
        </div>
        <div id="sub">
            <h2>宋词精选</h2>
            <ul>
```



```
                <li>…</li>
            </ul>
        </div>
    </div>
    <div id="sidebar">
        <h2>词人列表</h2>
        <ul>
            <li>…</li>
        </ul>
    </div>
</div>
```



设计页面能够自适应屏幕宽度，呈现不同的版式布局。当显示屏幕宽度在 999px 以上时，让 3 个栏目并列显示；当显示屏幕宽度在 639px 以上、1000px 以下时，设计两栏显示；当显示屏幕宽度在 640px 以下时，让 3 个栏目堆叠显示。

```
<style type="text/css">
/*默认样式*/
/*网页宽度固定，并居中显示*/
#container {width: 960px; margin: auto;}
/*主体宽度*/
#wrapper {width: 740px; float: left;}
/*设计 3 栏并列显示*/
#main {width: 520px; float: right;}
#sub {width: 200px; float: left;}
#sidebar { width: 200px; float: right;}
/*窗口宽度在 999px 以上*/
@media screen and (min-width: 1000px) {
    /*3 栏显示*/
    #container {width: 1000px;}
    #wrapper {width: 780px; float: left;}
    #main {width: 560px; float: right;}
    #sub {width: 200px; float: left; }
    #sidebar {width: 200px; float: right;}
}
/*窗口宽度在 639px 以上、1000px 以下*/
@media screen and (min-width: 640px) and (max-width: 999px) {
    /*两栏显示*/
    #container {width: 640px; }
    #wrapper {width: 640px; float: none;}
    .height {line-height: 300px; }
    #main {width: 420px; float: right;}
    #sub {width: 200px; float: left; }
    #sidebar {width: 100%; float: none;}
}
/*窗口宽度在 640px 以下*/
@media screen and (max-width: 639px) {
    /*1 栏显示*/
    #container {width: 100%;}
```





Note

```
        #wrapper {width: 100%; float: none;}
        #main {width: 100%; float: none;}
        #sub {width: 100%; float: none;}
        #sidebar {width: 100%; float: none;}
    }
</style>
```

当显示屏幕宽度在 999px 以上时，3 栏并列显示，预览效果如图 14.2 所示。

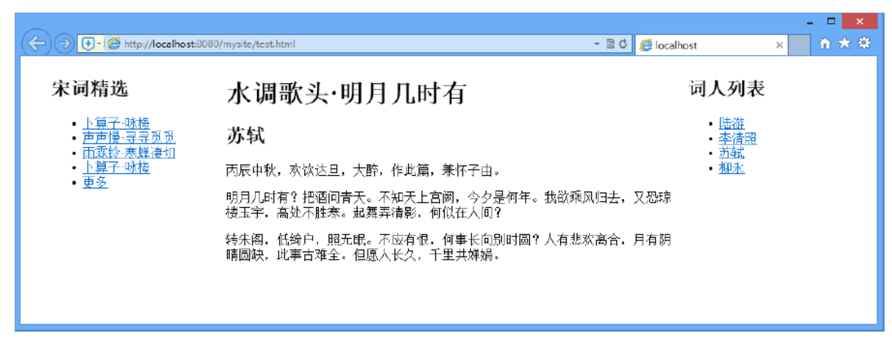


图 14.2　显示屏幕宽度在 999px 以上时页面显示效果

当显示屏幕宽度在 639px 以上、1000px 以下时，两栏显示，预览效果如图 14.3 所示；当显示屏幕宽度在 640px 以下时，3 个栏目从上往下堆叠显示，预览效果如图 14.4 所示。





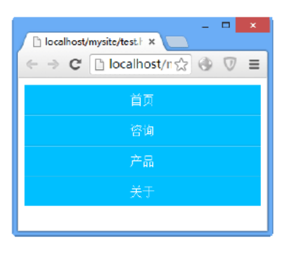
图 14.3　宽度在 639px 以上、1000px 以下时效果　　　图 14.4　宽度在 640px 以下时效果



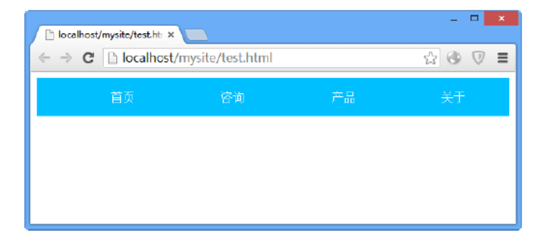
视 频 讲 解

14.2.3　设计响应式菜单

本例设计一个响应式菜单，能够根据设备显示不同的伸缩盒布局效果。在小屏设备上，从上到下显示；在默认状态下，从左到右显示，右对齐盒子；当设备小于 801px 时，设计导航项目分散对齐显示，预览效果如图 14.5 所示。

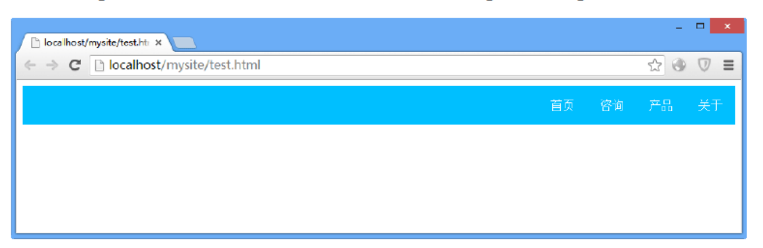


小于 601px 屏幕



介于 600px 和 800px 之间设备





大于 799px 屏幕

图 14.5　定义伸缩项目居中显示

主要代码如下：

```
<style type="text/css">
/*默认伸缩布局*/
.navigation {
    list-style: none;
    margin: 0;
    background: deepskyblue;
    /*启动伸缩盒布局*/
    display: -webkit-box;
    display: -moz-box;
    display: -ms-flexbox;
    display: -webkit-flex;
    display: flex;
    -webkit-flex-flow: row wrap;
    /*所有列面向主轴终点位置靠齐*/
    justify-content: flex-end;
}
/*设计导航条内超链接默认样式*/
.navigation a {text-decoration: none; display: block; padding: 1em; color: white;}
/*设计导航条内超链接在鼠标经过时的样式*/
.navigation a:hover {background: blue;}
/*在小于 801px 设备下伸缩布局*/
@media all and (max-width: 800px) {
    ./*当在中等屏幕中，导航项目居中显示，并且剩余空间平均分布在列表之间*/
    .navigation {justify-content: space-around;}}
/*在小于 601px 设备下伸缩布局*/
@media all and (max-width: 600px) {
    .navigation { /*在小屏幕下，没有足够空间行排列，可以换成列排列*/
        -webkit-flex-flow: column wrap;
```





Note

```
            flex-flow: column wrap;
            padding: 0;}
        .navigation a {
            text-align: center;
            padding: 10px;
            border-top: 1px solid rgba(255,255,255,0.3);
            border-bottom: 1px solid rgba(0,0,0,0.1);}
        .navigation li:last-of-type a {border-bottom: none;}
}
</style>
<ul class="navigation">
    <li><a href="#">首页</a></li>
    <li><a href="#">咨询</a></li>
    <li><a href="#">产品</a></li>
    <li><a href="#">关于</a></li>
</ul>
```

视频讲解

14.2.4 设计自动隐藏布局

本例设计一个响应式页面布局效果，并能根据显示屏幕宽度变化自动隐藏或调整版式显示。

【操作步骤】

第 1 步，新建 HTML5 文档，在头部<head>标签内定义视口信息。使用<meta>标签设置视口缩放比例为 1，让浏览器使用设备的宽度作为视图的宽度，并禁止初始缩放。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
```

第 2 步，IE8 或者更早的浏览器并不支持媒体查询，可以使用 media-queries.js 或者 respond.js 插件进行兼容。

```
<!--[if lt IE 9]>
    <script src="http://css3-mediaqueries-js.googlecode.com/svn/trunk/css3-mediaqueries.js"></script>
<![endif]-->
```

第 3 步，设计页面 HTML 结构。整个页面基本布局包括头部、内容、侧边栏和页脚。内容容器宽度是 600px，而侧边栏宽度是 300px，线框图如图 14.6 所示。

```
<div id="pagewrap">
    <div id="header">
        <h1>唐诗赏析</h1>
    </div>
    <div id="content">
        <h1>水调歌头 • 明月几时有</h1>
        <h2>苏轼</h2>
        <p>…</p>
```






```
        </div>
        <div id="sidebar">
            <h2>宋词精选</h2>
            <ul>
                <li>…</li>
            </ul>
        </div>
        <div id="footer">
            <h2>词人列表</h2>
            <ul>
                <li>…</li>
            </ul>
        </div>
    </div>
```



图 14.6　设计页面结构

第 4 步，使用 CSS3 媒体查询设计当视图宽度小于等于 980px 时，如下规则生效。将所有的容器宽度从像素值设置为百分比以使得容器大小自适应。

```
/*当窗口视图小于等于 980px 时响应下面样式*/
@media screen and (max-width: 980px) {
    #pagewrap {width: 94%;}
    #content {width: 65%;}
    #sidebar {width: 30%;}
}
```

第 5 步，为小于等于 700px 的视图指定<div id="content">和<div id="sidebar">的宽度为自适应，并且清除浮动，使得这些容器按全宽度显示。

```
/*当窗口视图小于等于 700px 时响应下面样式*/
@media screen and (max-width: 700px) {
    #content {
        width: auto;
        float: none;
```

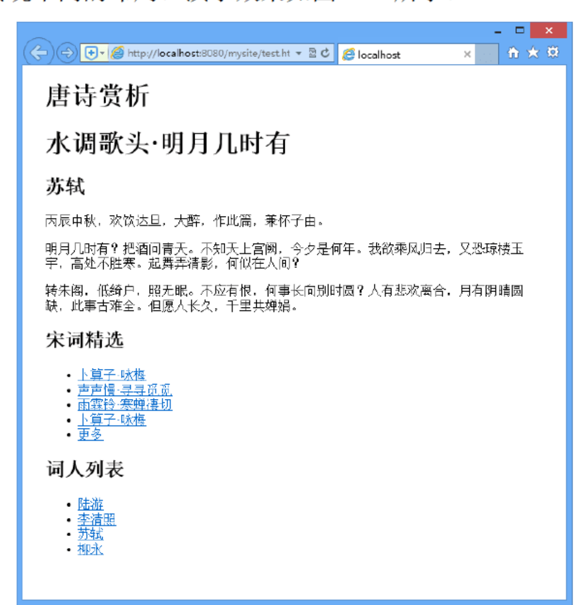


```
    }
    #sidebar {
        width: auto;
        float: none;
    }
}
```

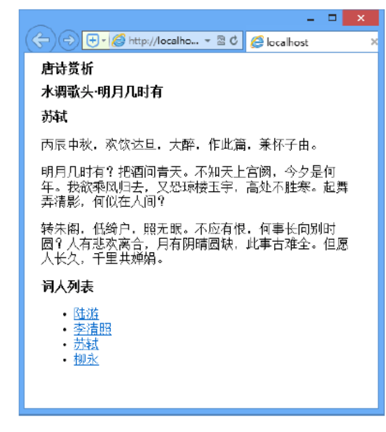
第 6 步，对于小于等于 480px（手机屏幕）的情况，将 h1 和 h2 的字体大小修改为 16px，并隐藏侧边栏<div id="sidebar">。

```
/*当窗口视图小于等于 480px 时响应下面样式*/
@media screen and (max-width: 480px) {
    h1, h2 {font-size: 16px;}
    #sidebar {display: none;}
}
```

第 7 步，可以根据需要添加更多媒体查询，目的在于为指定的视图宽度指定不同的 CSS 规则，来实现不同的布局。演示效果如图 14.7 所示。



平板屏幕下效果



手机屏幕下效果

图 14.7　设计不同宽度下的视图效果



视 频 讲 解

14.2.5　设计自适应手机页面

本例设计页面宽度为 980px，对于桌面屏幕来说，该宽度适用于任何大于 1024px 的分辨率。通过媒体查询监测宽度小于 980px 的设备，并将页面宽度由固定方式改为液态版式，布局元素的宽度随着浏览器窗口的尺寸变化进行调整。当可视部分的宽度进一步减小到 650px 以下时，主要内容部分的容器宽度会增大至全屏，而侧边栏将被置于主内容部分的下方，整个页面变为单列布局。演示效果如图 14.8 所示。









图 14.8　在不同宽度下的视图效果

【操作步骤】

第 1 步，新建 HTML5 文档，构建文档结构，包括页头、主要内容部分、侧边栏和页脚。

```
<div id="pagewrap">
    <header id="header">
        <hgroup>
            <h1 id="site-logo">网站 LOGO</h1>
            <h2 id="site-description">网站描述信息</h2>
        </hgroup>
        <nav>
            <ul id="main-nav">
                <li><a href="#">导航链接，可以扩展</a></li>
            </ul>
        </nav>
        <form id="searchform">
            <input type="search">
        </form>
    </header>
    <div id="content">
        <article class="post">主体内容区域</article>
    </div>
    <aside id="sidebar">
        <section class="widget"> 侧栏栏目</section>
    </aside>
    <footer id="footer">页脚区域</footer>
</div>
```

第 2 步，IE9 之前浏览器不支持 HTML5 标签，使用 html5.js 来帮助这些旧版本的 IE 浏览器创建 HTML5 元素节点。

```
<!--[if lt IE 9]>
<script src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
<![endif]-->
```



第 3 步，设计 HTML5 块级元素样式，将这些新元素声明为块级样式。

```
article, aside, details, figcaption, figure, footer, header, hgroup, menu, nav, section {display: block;}
```



第 4 步，设计主要结构的 CSS 样式。这里将注意力集中在整体布局上。整体设计在默认情况下页面容器的固定宽度为 980px；页头部分（header）的固定高度为 160px；主要内容部分（content）的宽度为 600px，左浮动；侧边栏（sidebar），宽度为 280px，右浮动。

```
<style type="text/css">
#pagewrap {
    width: 980px;
    margin: 0 auto;
}
#header {height: 160px;}
#content {
    width: 600px;
    float: left;
}
#sidebar {
    width: 280px;
    float: right;
}
#footer {clear: both;}
</style>
```

第 5 步，调用 css3-mediaqueries.js 文件，解决 IE8 及其以前版本不支持 CSS3 媒体查询的问题。

```
<!--[if lt IE 9]>
     <script src="http://css3-mediaqueries-js.googlecode.com/svn/trunk/css3-mediaqueries.js"></script>
<![endif]-->
```

第 6 步，创建 CSS 样式表，并在页面中调用。

```
<link href="media-queries.css" rel="stylesheet" type="text/css">
```

第 7 步，借助媒体查询设计自适应布局。当浏览器可视部分宽度大于 650px 小于 981px 时，将 pagewrap 的宽度设置为 95%，将 content 的宽度设置为 60%，将 sidebar 的宽度设置为 30%。

```
@media screen and (max-width: 980px) {
    #pagewrap {width: 95%;}
    #content {
        width: 60%;
        padding: 3% 4%;
    }
    #sidebar {width: 30%;}
    #sidebar .widget {
        padding: 8% 7%;
        margin-bottom: 10px;
    }
}
```

第 8 步，当浏览器可视部分宽度小于 651px 时，将 header 的高度设置为 auto；将 searchform 绝对定位在 top: 5px 的位置；将 main-nav、site-logo、site-description 的定位设置为 static；将 content 的宽度设置为 auto（主要内容部分的宽度将扩展至满屏），并取消 float 设置；将 sidebar 的宽度设置为 100%，



并取消 float 设置。

```
@media screen and (max-width: 650px) {
    #header {height: auto;}
    #searchform {
        position: absolute;
        top: 5px;
        right: 0;
    }
    #main-nav {position: static;}
    #site-logo {
        margin: 15px 100px 5px 0;
        position: static;
    }
    #site-description {
        margin: 0 0 15px;
        position: static;
    }
    #content {
        width: auto; margin: 20px 0;
        float: none;
    }
    #sidebar {
        width: 100%; margin: 0;
        float: none;
    }
}
```

第 9 步，当浏览器可视部分宽度小于 481px（480px 是传统手机横屏时的宽度）时，禁用 HTML 节点的字号自动调整。默认情况下，手机会将过小的字号放大，这里可以通过-webkit-text-size-adjust 属性进行调整，将 main-nav 中字号设置为 90%。

```
@media screen and (max-width: 480px) {
    html {-webkit-text-size-adjust: none;}
    #main-nav a {
        font-size: 90%;
        padding: 10px 8px;
    }
}
```

第 10 步，设计弹性图片。为图片设置 max-width: 100%和 height: auto，设计图像弹性显示。

```
img {
    max-width: 100%; height: auto;
    width: auto\9; /*兼容 IE8*/
}
```

第 11 步，设计弹性视频。对于视频也需要做 max-width: 100%的设置，但是 Safari 对 embed 的该属性支持不是很好，所以使用 width: 100%来代替。

```
.video embed, .video object, .video iframe {
    width: 100%; min-height: 300px;
```



```
        height: auto;
    }
```

第 12 步，在默认情况下，手机端 Safari 浏览器会对页面进行自动缩放，以适应屏幕尺寸。这里可以使用以下的 meta 设置，将设备的默认宽度作为页面在 Safari 的可视部分宽度，并禁止初始化缩放。



```
<meta name="viewport" content="width=device-width; initial-scale=1.0">
```

14.3 在线练习

本节专题练习响应式网页设计，同时复习 CSS3 的重要属性，强化训练应用 CSS3 新功能的能力。感兴趣的读者可以扫码练习。



在线练习 1



在线练习 2



第15章 使用 JavaScript 控制 CSS 样式

在网页设计中，经常会用 JavaScript 代码控制页面样式，这种样式也称为脚本样式，因此用户要了解 JavaScript 代码的基本用法和操作 CSS 样式的一般方法。本章将介绍如何使用 JavaScript 控制 CSS 样式来设计各种动态效果。

【学习重点】

- 了解使用 JavaScript 控制 CSS 样式的方法。
- 使用 JavaScript 控制网页对象的大小和显隐。
- 设计运动效果。
- 设计渐隐、渐显效果。
- 在网页中添加各种交互式响应或动态特效。



15.1　在网页中使用 JavaScript 脚本



Note



视频讲解

JavaScript 是目前最流行、应用最广泛的 Web 编程语言。一般情况下，JavaScript 代码只能够在网页中发挥作用，当然编写 JavaScript 代码的方法也很简单。

15.1.1　使用<script>标签

在 HTML 页面中嵌入 JavaScript 脚本需要使用<script>标签，用户可以直接在<script>标签中编写 JavaScript 代码，或者单独编写 JavaScript 文件，然后通过<script>标签导入。

【示例 1】直接在页面中嵌入 JavaScript 代码。

【操作步骤】

第 1 步，新建 HTML 文档，保存为 test.html，然后在<head>标签内插入一个<script>标签。

第 2 步，为<script>标签指定 type 属性值为"text/javascript"。现代浏览器默认<script>标签的类型为 JavaScript 脚本，因此省略 type 属性依然能够被正确执行，但是考虑到代码的兼容性，建议定义该属性。

第 3 步，直接在<script>标签内部输入 JavaScript 代码：

```
<script type="text/javascript">
function hi(){
    document.write("<h1>Hello,World!</h1>");
}
hi();
</script>
```

上面 JavaScript 脚本先定义了一个 hi()函数，该函数被调用后会在页面中显示字符“Hello,World!”。document 表示 DOM 网页文档对象，document.write()表示调用 document 对象的 write()方法，在当前网页源代码中写入 HTML 字符串"<h1>Hello,World!</h1>"。

调用 hi()函数，浏览器将在页面中显示一级标题字符“Hello,World!”。

第 4 步，保存网页文档，在浏览器中预览，则显示效果如图 15.1 所示。

包含在<script>标签内的 JavaScript 代码被浏览器从上至下依次解释执行。

【示例 2】包含外部 JavaScript 文件。

【操作步骤】

第 1 步，新建文本文件，保存为 test.js。注意，扩展名为.js，表示该文本文件是 JavaScript 类型的文件。

> 提示：使用<script>标签包含外部 JavaScript 文件时，默认文件类型为 JavaScript，因此.js 扩展名不是必需的，浏览器不会检查包含 JavaScript 的文件的扩展名。在高级开发中，使用 JSP、PHP 或其他服务器端语言动态生成 JavaScript 代码时可以使用任意扩展名，如果不使用.js 扩展名，用户应确保服务器能返回正确的 MIME 类型。

第 2 步，打开 test.js 文本文件，在其中编写下面代码，定义简单的输出函数。



```
function hi(){
    alert("Hello,World!");
}
```

在上面代码中，alert()表示 Window 对象的方法，调用该方法将弹出一个提示对话框，显示参数字符串“Hello,World!”。



第 3 步，保存 JavaScript 文件，注意与网页文件的位置关系。这里保存 JavaScript 文件的位置与调用该文件的网页文件位于相同目录下。

第 4 步，新建 HTML 文档，保存为 test1.html。然后在<head>标签内插入一个<script>标签。定义 src 属性，设置属性值为指向外部 JavaScript 文件的 URL 字符串。代码如下：

```
<script type="text/javascript" src="test.js"></script>
```

第 5 步，在上面<script>标签下一行继续插入一个<script>标签，直接在<script>标签内部输入 JavaScript 代码，调用外部 JavaScript 文件中的 hi()函数。

```
<script type="text/javascript" src="test.js"></script>
<script type="text/javascript">
hi();           //调用外部 JavaScript 文件的函数
</script>
```

第 6 步，保存网页文档，在浏览器中预览，则显示效果如图 15.2 所示。



图 15.1　第一个 JavaScript 程序

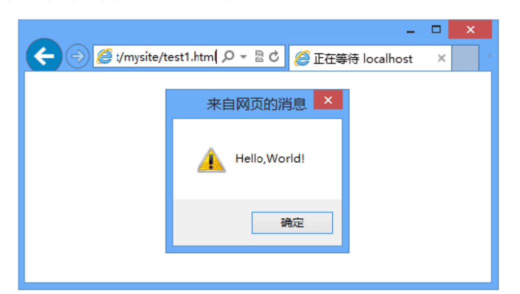


图 15.2　调用外部函数弹出提示对话框

提示： 定义 src 属性的<script>标签不应再包含 JavaScript 代码。如果嵌入了代码，则只会下载并执行外部 JavaScript 文件，嵌入代码会被忽略。src 属性可以包含外部域的 JavaScript 文件。例如：

```
<script type="text/javascript" src="http://www.sothersite.com/test.js"></script>
```

15.1.2　比较脚本样式与 CSS 样式



视 频 讲 解

JavaScript 代码与 CSS 代码不会相互干扰，但是由于 JavaScript 可以控制 CSS 样式，所以它们之间仍然存在某些关联。对于 CSS 文件来说，样式所引用的外部文件的路径都是以代码所在位置作为参考来进行设置的，而 JavaScript 恰恰相反，它是以所引用的网页位置作为参考进行设置的。

【示例】 有一个简单的站点结构，网页文件位于根目录，而 CSS 文件、JavaScript 文件和图像文件都位于根目录下 images 文件夹中，如图 15.3 所示。







Note

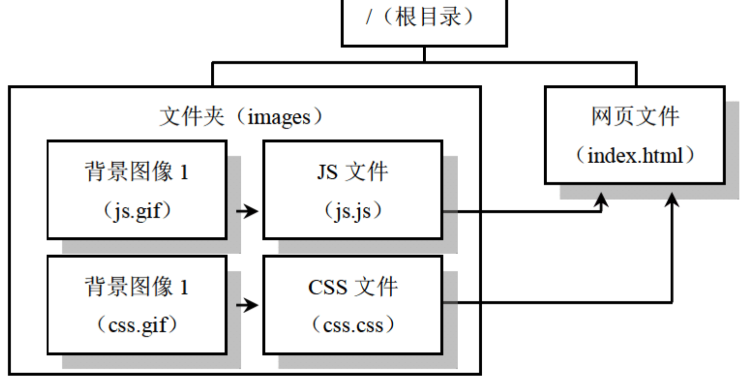


图 15.3　一个简单的站点结构

下面分别使用 CSS 样式和 JavaScript 脚本样式为网页中<div id="box">标签定义背景图像。

【操作步骤】

第 1 步，新建样式表文件，保存为 css.css，存放于 images 文件夹中。

第 2 步，在 CSS 样式表文件（css.css）中定义方法如下：

```
#box {
    background:url(css.gif);
}
```

CSS 文件与背景图像文件都在同一目录（images 文件夹）下，所以可以直接引用，而不用考虑网页文件的位置。

第 3 步，新建 JavaScript 文本，保存为 js.js，存放于 images 文件夹中。

第 4 步，在 js.js 文件中输入下面代码，使用 JavaScript 脚本定义<div id="box">的背景图像。

```
window.onload = function(){
    document.getElementById("box").style.backgroundImage="url(images/ js.gif)";
}
```

从上面代码可以看到，JavaScript 文件所引用的背景图像路径是以网页文件的位置为参考来进行设置的，而不用考虑 JavaScript 文件的具体位置，如果网页文件不动，则 JavaScript 文件所引用的路径是不会变化的。

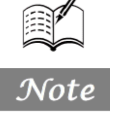
第 5 步，新建网页文件，保存为 index.html，存放于根目录下。

第 6 步，在网页文件中同时引用 CSS 和 JavaScript 文件。

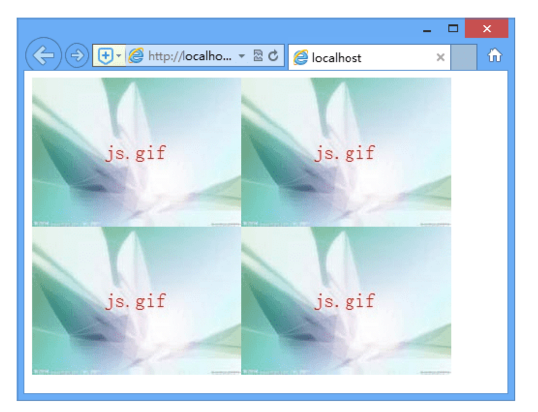
```
<style type="text/css">
#box {
    width:440px;
    height:312px;
}
</style>
<script type="text/javascript"  src="images/js.js"></script>
<link href="images/css.css" rel="stylesheet" type="text/css">

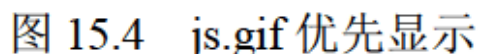
<div id="box"></div>
```





第 7 步，保存网页文档，在浏览器中预览，会发现<div id="box">标签显示 JavaScript 脚本定义的背景图像效果，如图 15.4 所示。




图 15.4　js.gif 优先显示

总之，JavaScript 文件与 CSS 文件中代码在引用外部图像文件时，相对路径设置是不同的，具体区别如下。

☑　CSS 文件：考虑 CSS 文件与导入的外部图像文件的位置关系。

☑　JavaScript 文件：考虑网页文件与导入的外部图像文件的位置关系。

另外，当同时使用 CSS 和 JavaScript 为页面对象定义样式时，JavaScript 脚本样式的优先级要高于 CSS 样式的优先级。

15.2　获取网页对象

使用 JavaScript 控制 CSS 样式的第一步是获取网页对象，以实现对其进行控制。

15.2.1　获取元素



为了获取文档结构中的元素节点，DOM 提供了两个方法。

1．使用 getElementById()方法

使用 getElementById()方法可以精确获取指定元素的引用指针。具体用法如下：

```
o = document.getElementById(ID)
```

其中 o 表示指定元素的引用指针，参数 ID 表示文档结构中对应元素的 id 属性值。如果文档中不存在指定元素，则返回值为 null。该方法只适用于 document 对象。

【示例 1】下面脚本能够获取对<div id="box">对象的控制权。

```
<div id="box">盒子</div>
<script>
var box = document.getElementById("box");    //获取 id 属性值为 box 的指定元素的引用指针
</script>
```



Note

getElementById()方法返回指定元素的对象，这个对象包含 nodeName、nodeType 等属性，简单说明如下。

☑ nodeName 表示节点的名称。如果是元素节点，则 nodeName 返回值为标签名称，标签名称永远是大写；如果是属性节点，则 nodeName 返回值为属性的名称；如果是文本节点，则 nodeName 返回值永远是#text 标识符；如果是文档节点，则 nodeName 返回值永远是#document 标识符。

☑ nodeType 表示节点的类型。该属性的返回值比较多，常用节点类型：1 表示元素类型，2 表示属性，3 表示文本，8 表示注释，9 表示文档。

【示例 2】在下面示例中，使用 getElementById()方法获取<div id="box">对象的引用指针，然后利用 nodeName、nodeType 属性查看该对象的节点名称和节点类型。

```
<div id="box">盒子</div>
<script>
var box = document.getElementById("box");          //获取指定盒子的引用指针
var info = "nodeName：" + box.nodeName;             //获取该节点的名称
info += "\rnodeType：" + box.nodeType;              //获取该节点的类型
alert(info);                                       //显示提示信息
</script>
```

2．使用 getElementByTagName()方法

使用 getElementByTagName()方法获取指定标签名称的所有元素对象。其用法如下：

```
a = document.getElementsByTagName(tagName)
```

其中参数 tagName 表示指定名称的标签，该方法返回值为一个元素集合。使用 length 属性可以获取集合中包含元素的个数，利用数组下标可以确定其中某个元素对象。

【示例 3】对于 DOM 元素集合来说，由于它们都是节点对象，因此可以使用 nodeName、nodeType 属性查看该对象的节点名称和节点类型。

```
<p id="p1">段落文本 1</p>
<p id="p2">段落文本 2</p>
<p id="p3">段落文本 3</p>
<script>
var p = document.getElementsByTagName("p");        //获取文档中所有 p 元素
alert(p[2].nodeName);                              //显示第 3 个 p 元素对象的节点名称
</script>
```

在实际开发中，常用 for 循环遍历集合中所有元素。

【示例 4】下面的代码使用 for 结构遍历获得的所有 p 元素，并设置 p 元素的 class 属性为 red。

```
<p id="p1">段落文本 1</p>
<p id="p2">段落文本 2</p>
<p id="p3">段落文本 3</p>
<script>
var p = document.getElementsByTagName("p");        //获取文档中所有 p 元素
for(var i=0;i<p.length;i++){                       //遍历 p 数据集合
    p[i].setAttribute("class","red");              //为每个 p 元素添加 class 类
}
</script>
```







提示：使用 document.getElementsByTagName("*")方式获取文档中所有元素节点的方法很少用，同时 IE 6.0 及其以下版本浏览器对其支持不是很好。对于 IE 浏览器来说，可以通过 document.all 来获取文档中所有元素节点。

15.2.2　使用 CSS 选择器匹配元素

HTML5 引入了与 jQuery 选择器相似的 DOM API 模块，该模块中的 querySelector()和 querySelectorAll()方法能够根据 CSS 选择器规范，便捷定位文档中指定元素。目前主流浏览器，包括 IE8+、Firefox、Chrome、Safari、Opera 均支持它们。

querySelector()和 querySelectorAll()方法的参数必须是符合 CSS 选择器规范的字符串，不同的是 querySelector()方法返回的是一个元素对象，querySelectorAll()方法返回的是一个元素集合。

【示例 1】 新建网页文档，输入下面的 HTML 结构代码。

```
<div class="content">
    <ul>
        <li>首页</li>
        <li class="red">财经</li>
        <li class="blue">娱乐</li>
        <li class="red">时尚</li>
        <li class="blue">互联网</li>
    </ul>
</div>
```

如果要获得第一个 li 元素，可以使用如下方法：

```
document.querySelector(".content ul li");
```

如果要获得所有 li 元素，可以使用如下方法：

```
document.querySelectorAll(".content ul li");
```

如果要获得所有 class 为 red 的 li 元素，可以使用如下方法：

```
document.querySelectorAll("li.red");
```

提示：DOM API 模块也包含 getElementsByClassName()方法，使用该方法可以获取指定类名的元素。例如：

```
document.getElementsByClassName("red");
```

注意：getElementsByClassName()方法只能够接收字符串，且为类名，而不需要加点号前缀，如果没有匹配到任何元素则返回空数组。

CSS 选择器是一个便捷的确定元素的方法，这是因为大家已经对 CSS 很熟悉了。当需要联合查询时，使用 querySelectorAll()更加便利。

【示例 2】 在文档中，一些 li 元素的 class 名称是 red，另一些元素的 class 名称是 blue，可以用 querySelectorAll()方法一次性获得这两类节点。

```
var lis = document.querySelectorAll("li.red, li.blue");
```

如果不使用 querySelectorAll()方法，那么要获得同样列表，需要更多工作。一个办法是选择所有



的 li 元素，然后通过迭代操作过滤出不需要的列表项目。

```
var result = [], lis1 = document.getElementsByTagName('li'), classname = '';
for(var i = 0, len = lis1.length; i < len; i++) {
    classname = lis1[i].className;
    if(classname === 'red' || classname === 'blue') {
        result.push(lis1[i]);
    }
}
```



Note

比较上面两种不同的用法，使用选择器 querySelectorAll()方法比使用 getElementsByTagName()要快很多。因此，如果浏览器支持 document.querySelectorAll()，那么最好使用它。

15.3 操作类样式

使用 JavaScript 控制 CSS 样式最简单、最直接的方法是为元素添加或删除类样式。



视频讲解

15.3.1 获取类样式

DOM 定义 getAttribute()方法可以获取指定元素的属性。其用法比较简单，只要指定元素及它的属性，即可快速反馈该元素所对应的属性值。

【示例 1】下面示例能够获取红色盒子和蓝色盒子，并显示这些元素所包含的 class 属性值。

```
<script>
window.onload = function() {
    var red = document.getElementById("red");            //获取红色盒子
    alert(red.getAttribute("class"));                    //显示红色盒子的 class 属性值
    var blue = document.getElementById("blue");          //获取蓝色盒子
    alert(blue.getAttribute("class"));                   //显示蓝色盒子的 class 属性值
}
</script>
<div id="red" class="red">红盒子</div>
<div id="blue" class="blue">蓝盒子</div
```

所传递的参数是一个字符串形式的元素属性名称，返回的是一个字符串类型的值，如果给定属性不存在，则返回的值为 null。

【示例 2】除了标准读取属性的方法外，HTML DOM 模型还支持快捷读取属性的方法。

```
window.onload = function() {
    var red = document.getElementById("red");
    alert(red.id);
    var blue = document.getElementById("blue");
    alert(blue.id);
}
```

对于 class 属性，必须使用 className 属性来读取，因为 class 是 JavaScript 保留字。同样，要读取 for 属性，则必须使用 htmlFor 属性名，这与 CSS 脚本中 float 和 text 属性被改名为 cssFloat 和 cssText 原因相同。

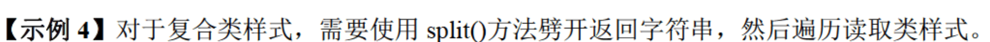




Note

【示例 3】使用 className 读取类样式。

```
<script>
window.onload = function() {
    var red = document.getElementById("red");           //获取红色盒子
    alert(red.className);                               //显示红色盒子的 class 属性值
    var blue = document.getElementById("blue");         //获取蓝色盒子
    alert(blue.className);                              //显示蓝色盒子的 class 属性值
}
</script>
<div id="red" class="red">红盒子</div>
<div id="blue" class="blue">蓝盒子</div
```



【示例 4】对于复合类样式，需要使用 split()方法劈开返回字符串，然后遍历读取类样式。

```
<script>
window.onload = function() {
    //所有类名生成的数组
    var classNameArray = document.getElementById("red").className.split(" ");
    for(var i in classNameArray ){                      //遍历数组
        alert(classNameArray[i]);                       //当前 class 名
    }
}
</script>
<div id="red" class="red blue">红盒子</div>
```

15.3.2　添加类样式



视 频 讲 解

为元素设置属性可以使用 setAttribute()方法实现，用法如下：

```
e.setAttribute(name,value)
```

其中参数 e 表示指定的元素对象，参数 name 和 value 分别表示属性名称和属性值。属性名和属性值必须以字符串的形式进行传递。如果元素中存在指定的属性，它的值将被刷新；如果不存在，则 setAttribute()方法将为元素创建该属性并赋值。

【示例 1】下面示例分别为页面中 div 元素设置 class 属性。

```
<script>
window.onload = function() {
    var red = document.getElementById("red");
    var blue = document.getElementById("blue");
    red.setAttribute("class", "red");
    blue.setAttribute("class", "blue");
}
</script>
<div id="red">红盒子</div>
<div id="blue">蓝盒子</div>
```

【示例 2】使用 setAttribute()方法存在弊端，一般通过 className 设置元素的类名。

```
<script>
window.onload = function() {
```



```
        var red = document.getElementById("red");
        var blue = document.getElementById("blue");
        red.className = "red";
        blue.className = "blue";
    }
    </script>
    <div id="red">红盒子</div>
    <div id="blue">蓝盒子</div>
```

Note

【示例 3】直接使用 className 添加类样式，会覆盖元素原来的类样式。可以采用叠加的方式添加类。

```
    <script>
    window.onload = function() {
        var red = document.getElementById("red");
        red.className = "red";
        red.className +=   " blue";
    }
    </script>
    <div id="red">红盒子</div>
```

【示例 4】使用叠加的方式添加类也存在问题，即容易添加大量重复的类。为此，定义一个检测函数，判断元素是否包含指定的类，然后决定是否添加类。

```
    <script>
    function hasClass(element,className){                    //类名检测函数
        var reg =new RegExp('(\\s|^)'+ className + '(\\s|$)');
        return   reg.test(element.className);                //使用正则表达式检测是否有相同的样式
    }
    function addClass(element,className){                    //添加类名函数
        if(!hasClass(element, className))
            element.className +=' ' + className;
    }
    window.onload = function() {
        var red = document.getElementById("red");
        addClass(red,'red');
        addClass(red,'blue');
    }
    </script>
    <div id="red">红盒子</div>
```


视频讲解

15.3.3 删除类样式

DOM 使用 removeAttribute()方法删除指定的属性，用法如下：

```
e.removeAttribute(name)
```

其中 e 表示一个元素对象，而参数 name 表示元素的属性名。

【示例 1】下面示例演示如何动态设置表格的边框。

```
    <script>
    window.onload = function() {//绑定页面加载完毕时的事件处理函数
```



```
        var table = document.getElementsByTagName("table")[0];          //获取表格外框的引用指针
        var del = document.getElementById("del");                       //获取“删除”按钮的引用指针
        var reset = document.getElementById("reset");                   //获取“恢复”按钮的引用指针
        del.onclick = function(){                                       //为“删除”按钮绑定事件处理函数
            table.removeAttribute("border");                            //移除边框属性
        }
        reset.onclick = function(){                                     //为“恢复”按钮绑定事件处理函数
            table.setAttribute("border", "2");                          //设置表格的边框属性
        }
    }
    </script>
    <table width="100%" border="2">
        <tr>
            <td>数据表格</td>
        </tr>
    </table>
    <button id="del">删除</button><button id="reset">恢复</button>
```



在上面示例中，设计了两个按钮，并分别绑定不同的事件处理函数。单击“删除”按钮即可调用表格的 removeAttribute()方法清除表格边框，单击“恢复”按钮即可调用表格的 setAttribute()方法重新设置表格边框的粗细。

【示例 2】下面示例演示如何自定义删除类函数，并调用该函数删除指定类名。

```
    <script type="text/javascript">
    function hasClass(element,className){//类名检测函数
        var reg =new RegExp('(\\s|^)'+ className + '(\\s|$)');
        return   reg.test(element.className);                           //使用正则表达式检测是否有相同的样式
    }function deleteClass(element,className){
         f(hasClass(element,className)){
            element.className.replace(reg,' '); //利用正则表达式捕获要删除的样式的名称，然后把它替换成一个
空白字符串，就相当于删除了
        }
    }
    window.onload = function() {
        var red = document.getElementById("red");
        deleteClass(red,'blue');
    }
    </script>
    <div id="red" class="red   blue   bold">红盒子</div>
```

上面的代码使用正则表达式检测 className 属性值字符串中是否包含指定的类名，如果存在，则使用空字符替换匹配到的子字符串，从而实现删除类名的目的。

15.4　操作 CSS 样式

在 JavaScript 脚本中获取页面元素之后，就可以使用 style 属性获取该元素的 CSS2Properties 对象。CSS2Properties 包含该对象的所有 CSS 脚本属性。设置这些属性与设置 CSS 样式的效果是一样的。



视 频 讲 解



Note

15.4.1 使用 style 对象

DOM 定义每个元素都继承一个 style 对象，style 对象包含一些方法，利用这些方法可以与 CSS 样式实现交互。但是，style 对象针对的是行内样式，不支持操作样式表，包括内部样式表（<style>标签包含的样式）或外部样式表。

1．getPropertyValue()方法

getPropertyValue()方法能够获取指定元素样式属性的值。具体用法如下：

```
var value = e.style.getPropertyValue(propertyName)
```

参数 propertyName 表示 CSS 属性名，不是 CSS 脚本属性名，对于复合名应该使用连字符进行连接。

【示例 1】下面代码使用 getPropertyValue()方法获取行内样式中 width 属性值，然后输出到盒子内显示，如图 15.5 所示。

```
<script>
window.onload = function(){
    var box = document.getElementById("box");           //获取<div id="box">
    var width = box.style.getPropertyValue("width");    //读取 div 元素的 width 属性值
    box.innerHTML =  "盒子宽度：" + width;               //输出显示 width 值
}
</script>

<div id="box" style="width:300px; height:200px;border:solid 1px red" >盒子</div>
```

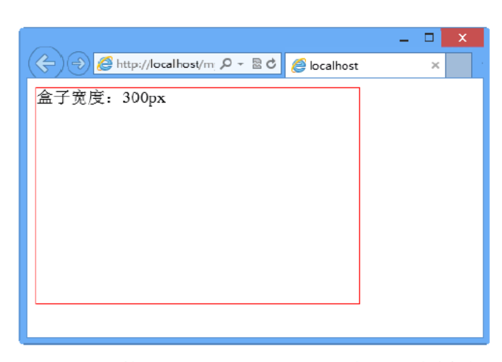


图 15.5　使用 getPropertyValue()读取行内样式

早期 IE 版本不支持 getPropertyValue()方法，但是可以通过 style 对象直接访问样式属性来获取指定样式的属性值。

提示：在设置 CSS 脚本属性时，应注意几个问题：

☑ 由于 float 是 JavaScript 保留字，禁止用户使用，因此，CSS2Properties 内没有与 float 属性对应的名称。为了解决这个问题，CSS2Properties 在 float 属性前增加了 css 前缀，使用 cssFloat 名来表示脚本中 float 属性。

☑ 在 CSS 中读写属性值时，不需要考虑值的类型。但是在 JavaScript 中，CSS2Properties 对象认定所有 CSS 属性值都是字符串，因此脚本中所有属性值都必须加上引号，以表示为字符串数据类型。例如：





Note

```
elementNode.style.fontFamily = "Arial, Helvetica, sans-serif";
elementNode.style.cssFloat = "left";
elementNode.style.color = "#ff0000";
```

☑ 在 CSS 样式中声明尾部的分号不能够作为属性值的一部分被引用，脚本中的分号只是 JavaScript 语法规则的一部分，而不是 CSS 声明中分号的引用。

☑ 声明中属性值所包含的单位等都必须作为值的一部分，完整地传递给 CSS 脚本属性，省略单位则所设置的脚本样式无效。例如:

```
elementNode.style.width = "100px";
```

☑ 在脚本中可以动态设置属性值，但最终赋值给属性的值应是一个字符串，且必须包含单位。例如:

```
elementNode.style.top = top + "px";
elementNode.style.right = right + "px";
elementNode.style.bottom = bottom + "px";
elementNode.style.left = left + "px";
```

【示例 2】针对示例 1 代码，可以使用如下方式读取 width 属性值。

```
<script>
window.onload = function(){
    var box = document.getElementById("box");
    var width = box.style.width;
    box.innerHTML =  "盒子宽度：" + width;
}
</script>
```

2．setProperty()方法

setProperty()方法可以为指定元素设置样式。具体用法如下：

```
e.style.setProperty(propertyName, value, priority)
```

参数说明如下。

☑ propertyName：设置 CSS 属性名。

☑ value：设置 CSS 属性值，包含属性值的单位。

☑ priority：表示是否设置!important 优先级命令，如果不设置可以以空字符串表示。

【示例 3】在下面示例中，使用 setProperty()方法定义盒子的显示宽度和高度分别为 400px 和 200px。

```
<script>
window.onload = function(){
    var box = document.getElementById("box");             //获取<div id="box">
    box.style.setProperty("width","400px","");            //定义盒子宽度为 400px
    box.style.setProperty("height","200px","");           //定义盒子高度为 200px
}
</script>

<div id="box" style="border:solid 1px red" >盒子</div>
```

如果要兼容早期 IE 浏览器，则可以使用如下方式设置。



Note

```
<script>
window.onload = function(){
    var box = document.getElementById("box");
    box.style.width = "400px";
    box.style.height = "200px";
}
</script>
```

3．removeProperty()方法

removeProperty()方法可以移除指定 CSS 属性的样式声明。具体用法如下：

```
e.style. removeProperty(propertyName)
```

4．item()方法

item()方法返回 style 对象中指定索引位置的 CSS 属性名称。具体用法如下：

```
var name = e.style.item(index)
```

参数 index 表示 CSS 样式的索引号。

5．getPropertyPriority()方法

getPropertyPriority()方法可以获取指定 CSS 属性中是否附加了!important 优先级命令，如果存在则返回 important 字符串，否则返回空字符串。

【示例 4】在下面示例中，定义鼠标移过盒子时，盒子的背景色为蓝色，而边框颜色为红色，当移出盒子时，又恢复到盒子默认设置的样式；而单击盒子时，则在盒子内输出动态信息，显示当前盒子的宽度和高度，演示效果如图 15.6 所示。

```
<script>
window.onload = function(){
    var box = document.getElementById("box");                    //获取盒子的引用
    box.onmouseover = function(){                                //定义鼠标经过时的事件处理函数
        box.style.setProperty("background-color", "blue", "");   //设置背景色为蓝色
        box.style.setProperty("border", "solid 50px red", "");   //设置边框为 50px 的红色实线
    }
    box.onclick = function(){                                    //定义鼠标单击时的事件处理函数
        box .innerHTML = (box.style.item(0) + ":" + box.style.getPropertyValue("width"));
                //显示盒子的宽度
        box .innerHTML = box .innerHTML + "<br>" +  (box.style.item(1) + ":" + box.style.getPropertyValue
("height"));                                                     //显示盒子的高度
    }
    box.onmouseout = function(){                                 //定义鼠标移出时的事件处理函数
        box.style.setProperty("background-color", "red", "");    //设置背景色为红色
        box.style.setProperty("border", "solid 50px blue", "");  //设置边框为 50px 的蓝色实线
    }
}
</script>

<div id="box" style="width:100px; height:100px; background-color:red; border:solid 50px blue;"></div>
```

【示例 5】针对示例 4，下面使用一种快捷方式设计相同的交互效果，这样能够兼容 IE 早期版本，

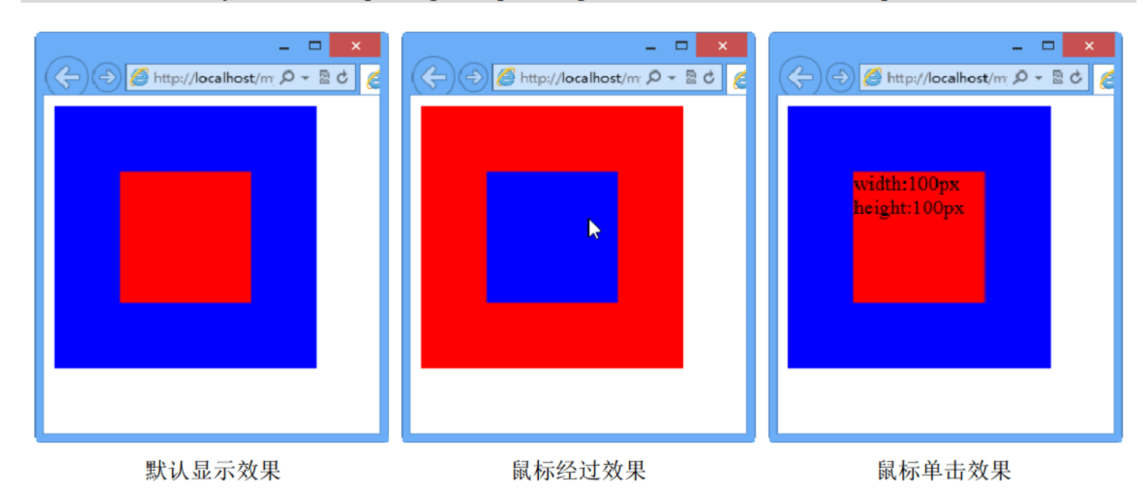


Note

页面代码如下：

```
<script>
window.onload = function(){
    var box = document.getElementById("box");          //获取盒子的引用
    box.onmouseover = function(){
        box.style.backgroundColor = "blue";            //设置背景样式
        box.style.border = "solid 50px red";           //设置边框样式
    }
    box.onclick = function(){                          //读取并输出行内样式
        box .innerHTML = "width:" + box.style.width;
        box .innerHTML = box .innerHTML + "<br>" +   "height:" + box.style.height;
    }
    box.onmouseout = function(){                       //设计鼠标移出之后，恢复默认样式
        box.style.backgroundColor = "red";
        box.style.border = "solid 50px blue";
    }
}
</script>

<div id="box" style="width:100px; height:100px; background-color:red; border:solid 50px blue;"></div>
```



默认显示效果　　　　鼠标经过效果　　　　鼠标单击效果

图 15.6　设计动态交互样式效果

【拓展】

非 IE 浏览器也支持 style 快捷访问方式，但是它无法获取 style 对象中指定序号位置的属性名称，此时可以使用 cssText 属性读取全部 style 属性值，再借助 JavaScript 方法把返回字符串劈开为数组。

【示例 6】 在下面示例中，使用 cssText 读取全部行内样式字符串，然后使用 String 的 split()方法把字符串劈开为数组，使用 for in 语句遍历数组，逐一读取每个样式，再使用 split()方法劈开属性和属性名，最后格式化输出显示，演示效果如图 15.7 所示。

```
<script>
window.onload = function(){
    var box = document.getElementById("box");          //获取盒子的引用
    var str = box.style.cssText;                       //读取盒子全部行内样式
```



Note

```
        var a = str.split(";");                            //把行内样式字符串转换为数组
        var temp="";
        for(var b in a){                                   //遍历行内样式
            var prop = a[b].split(":");                    //把每个样式字符串劈开为数组
            if(prop[0])                                    //如果存在属性，则输出显示
                temp += b + " ：" + prop[0] + " = " + prop[1] + "<br>";
        }
        box.innerHTML = "box.style.cssText = " + str;
        box.innerHTML = box.innerHTML + "<br><br>" + temp;  //把格式化后的行内样式输出显示
    }
    </script>

    <div id="box" style="width:600px; height:200px; background-color:#81F9A5; border:solid 2px blue;padding:
10px"></div>
```

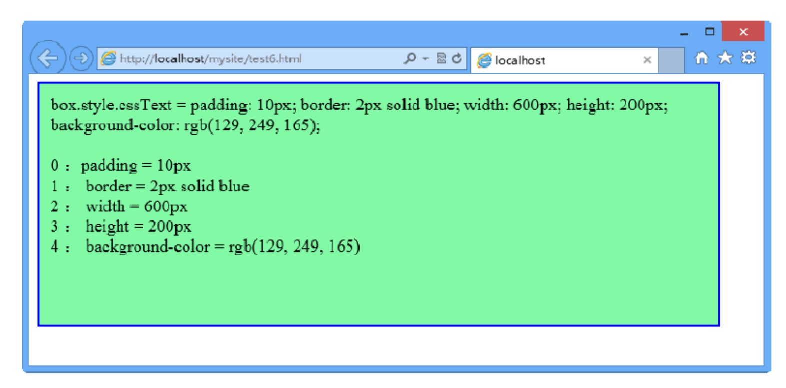


图 15.7　使用 cssText 属性获取行内样式

使用 getAttribute()方法也可以获取 style 属性值，不过该方法返回值保留 style 属性值的原始模样，而 cssText 属性返回值可能经过浏览器处理，且不同浏览器返回值格式略有不同。

【示例 7】修改示例 6 的代码，使用 getAttribute()方法获取行内样式字符串信息。

```
    <script>
    window.onload = function(){
        var box = document.getElementById("box");
        var str = box.getAttribute("style");
        var a = str.split(";");
        var temp="";
        for(var b in a){
            var prop = a[b].split(":");
        if(prop[0])
            temp += b + " ：" + prop[0] + " = " + prop[1] + "<br>";
        }
        box.innerHTML = "box.style.cssText = " + str;
        box.innerHTML = box.innerHTML + "<br><br>" + temp;
    }
    </script>

    <div id="box" style="width:600px; height:200px; background-color:#81F9A5; border:solid 2px blue;padding:
10px"></div>
```







15.4.2　使用 styleSheets 对象

document 对象包含一个 styleSheets 属性集合，它保存了文档中所有的样式表，包括内部样式表和外部样式表。

styleSheets 为每个样式表定义了一个 cssRules 对象，用来包含指定样式表中所有的规则（样式）。但是 IE 不支持 cssRules 对象，而预定义了 rules 对象表示样式表中的规则。

为了兼容主流浏览器，在使用前应该检测用户所使用浏览器的类型，以便调用不同的对象：

```
var cssRules = document.styleSheets[0].cssRules || document.styleSheets[0].rules;
```

在上面代码中，先判断浏览器是否支持 cssRules 对象，如果支持则使用 cssRules（非 IE 浏览器），否则使用 rules（IE 浏览器）。

【示例】在下面示例中，通过<style>标签定义一个内部样式表，为页面中的<div id="box">标签定义 4 个属性：宽度、高度、背景色和边框。然后在脚本中使用 styleSheets 访问这个内部样式表，把样式表中的第一个样式的所有规则读取出来，在盒子中输出显示，如图 15.8 所示。

```
<style type="text/css">
#box {
    width: 400px;
    height: 200px;
    background-color:#BFFB8F;
    border: solid 1px blue;
}
</style>
<script>
window.onload = function(){
    var box = document.getElementById("box");
    var cssRules = document.styleSheets[0].cssRules || document.styleSheets[0].rules;//判断浏览器类型
    box.innerHTML =  "<h3>盒子样式</h3>"
    box.innerHTML +=  "<br>边框：" + cssRules[0].style.border;          //读取 cssRules 的 border 属性
    box.innerHTML +=  "<br>背景：" + cssRules[0].style.backgroundColor;  //读取 cssRules 的 background-
color 属性
    box.innerHTML +=  "<br>高度：" + cssRules[0].style.height;          //读取 cssRules 的 height 属性
    box.innerHTML +=  "<br>宽度：" + cssRules[0].style.width;           //读取 cssRules 的 width 属性
}
</script>

<div id="box"></div>
```

提示：cssRules（或 rules）的 style 对象在访问 CSS 属性时，使用的是 CSS 脚本属性名，因此所有属性名称中不能使用连字符。例如：

```
cssRules[0].style.backgroundColor;
```

这与行内样式中的 style 对象的 setProperty()方法不同，setProperty()方法使用的是 CSS 属性名。例如：

```
box.style.setProperty("background-color", "blue", "");
```





Note

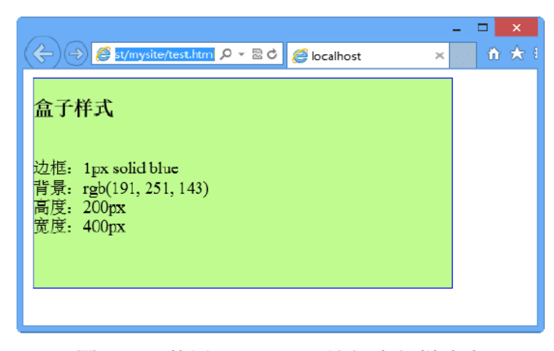


图 15.8　使用 styleSheets 访问内部样式表



视 频 讲 解

15.4.3　访问样式

styleSheets 包含文档中所有样式表，用户可以通过下标访问每个样式表，每个数组元素代表一个样式表，数组的索引位置是由样式表在文档中的位置决定的。每个<style>标签包含的所有样式表示一个内部样式表，每个独立的 CSS 文件表示一个外部样式表。

【示例】下面示例演示如何准确找到指定样式表中的样式属性。

【操作步骤】

第 1 步，启动 Dreamweaver，新建 CSS 文件，保存为 style1.css，存放在根目录下。

第 2 步，在 style1.css 中输入下面样式代码，定义一个外部样式表。

```
@charset "utf-8";
body {color:black;}
p {color:gray;}
div {color:white;}
```

第 3 步，新建 HTML 文档，命名为 test.html，保存在根目录下。

第 4 步，使用<style>标签定义一个内部样式表，设计如下样式。

```
<style type="text/css">
#box {color:green;}
.red {color:red;}
.blue {color:blue;}
</style>
```

第 5 步，使用<link>标签导入外部样式表文件 style1.css。

```
<link href="style1.css" rel="stylesheet" type="text/css" media="all" />
```

第 6 步，在文档中插入一个<div id="box">标签。

```
<div id="box"></div>
```

第 7 步，使用<script>标签在头部位置插入一段脚本。设计在页面初始化完毕后，使用 styleSheets 访问文档中第二个样式表，然后访问该样式表中第一个样式的 color 属性。

```
<script>
window.onload = function(){
    var cssRules = document.styleSheets[1].cssRules || document.styleSheets[1].rules;
```



```
        var box = document.getElementById("box");
        box.innerHTML =  "第二个样式表中第一个样式的 color 属性值 = " + cssRules[0].style.color;
    }
    </script>
```



第 8 步，保存页面，整个文档的代码如下：

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<style type="text/css">
#box {color:green;}
.red {color:red;}
.blue {color:blue;}
</style>
<link href="style1.css" rel="stylesheet" type="text/css" media="all" />
<script>
window.onload = function(){
    var cssRules = document.styleSheets[1].cssRules || document.styleSheets[1].rules;
    var box = document.getElementById("box");
    box.innerHTML =  "第二个样式表中第一个样式的 color 属性值 = " + cssRules[0].style.color;
}
</script>
</head>
<body>
<div id="box"></div>
</body>
</html>
```

最后，在浏览器中预览页面，则可以看到访问的 color 属性值为 black，如图 15.9 所示。

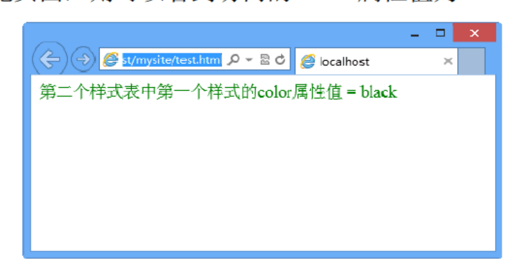


图 15.9　使用 styleSheets 访问外部样式表

提示： 上面示例中，styleSheets[1]表示外部样式表文件（style1.css），而 cssRules[0]表示外部样式表文件中的第一个样式。cssRules[0].style.color 可以获取外部样式表文件中第一个样式的 color 属性的声明值。反之，如果把<link>标签放置在内部样式表的上面，即代码如下：

```
<head>
<link href="style1.css" rel="stylesheet" type="text/css" media="all" />
<style type="text/css">
#box {color:green;}
.red {color:red;}
```



```
.blue {color:blue;}
</style>
</head>
```



Note

则上面脚本将返回内部样式表中第一个样式的 color 属性值，即 green。如果把外部样式表转换为内部样式表，或者把内部样式表转换为外部样式表文件，不会影响 styleSheets 的访问。因此，样式表和样式的索引位置是不受样式表类型和样式的选择符限制的。任何类型的样式表（不管是内部的，还是外部的）都在同一个平台上按在文档中解析位置进行索引。同理，不同类型选择符的样式在同一个样式表中也是根据先后位置进行索引。



视频讲解

15.4.4 编辑样式

cssRules 的 style 对象不仅可以访问属性，还可以设置属性值。

【示例】在下面示例中，样式表中包含 3 个样式，其中蓝色样式类（.blue）定义字体显示为蓝色。利用脚本修改该样式类（.blue 规则）字体颜色显示为浅灰色（#999），最后显示效果如图 15.10 所示。

```
<style type="text/css">
#box {color:green;}
.red {color:red;}
.blue {color:blue;}
</style>
<script>
window.onload = function(){
    var cssRules = document.styleSheets[0].cssRules || document.styleSheets[0].rules;
    cssRules[2].style.color="#999";      //修改样式表中指定属性的值
}
</script>

<p class="blue">原为蓝色字体，现在显示为浅灰色。</p>
```



图 15.10　修改样式表中的样式

提示：使用上述方法修改样式表中的类样式，会影响其他对象或文档对当前样式表的引用，因此在使用时请务必谨慎。

15.5　案例实战

本节将通过多个案例帮助读者上机练习使用 JavaScript 控制 CSS 的方法。







15.5.1　设计显示和隐藏

简单的隐藏元素可以通过 style.display 属性来实现。

【示例 1】下面示例能够遍历结构中所有的 p 元素，并把 class 属性值不为 main 的段落文本全部隐藏。

```
<p>p1</p>
<p class="main">p2</p>
<p>p3</p>
<script>
var p = document.getElementsByTagName("p");
for(var i = 0; i < p.length; i ++ ){
    if(p[i].className == "main") continue;
    //如果 class 属性值为 main，则跳过
    p[i].style.display = "none";                //隐藏元素
}
</script>
```

恢复 style.display 属性的默认值，只需设置 style.display 属性值为空字符串（style.display = ""）。

【示例 2】由于显示和隐藏是交互设计中经常用到的技巧，所以有必要对其进行功能封装，以实现代码重用和灵活应用，并能够兼容不同浏览器。

当指定元素和布尔值参数时，则元素能够根据布尔值 true 或 false 决定是否进行显示或隐藏，如果不指定第二个布尔值参数，则函数将对元素进行显示或隐藏切换。

```
//设置或切换元素的显示或隐藏
//参数：e 表示要显示或隐藏的元素，b 是一个布尔值，当为 true 时，将显示元素 e；
当为 false 时，将隐藏元素 e。如果省略参数 b，则根据元素 e 的显示状态，进行显示或隐藏切换
//返回值：无
function display(e, b){
    //监测第二个参数的类型。如果该参数存在且不为布尔值，则抛出异常
    if(b && (typeof b != "boolean")) throw new Error("第二个参数应该是布尔值!");
    var c = getStyle(e, "display");             //获取当前元素的显示属性值
    (c != "none") && (e._display = c);          //记录元素的显示性质，并存储到元素的属性中
    e._display = e._display || "";              //如果没有定义显示性质，则赋值为空字符串
    if(b || (c == "none") ){                    //当第二个参数值为 true 或者元素隐藏时
        e.style.display = e._display;           //将调用元素的_display 属性值恢复元素或显示元素
    }
    else{
        e.style.display = "none";               //否则隐藏元素
    }
}
```

下面在页面中设置一个向右浮动的元素 p。连续调用 3 次 display()函数后，则相当于隐藏元素，代码如下：

```
<p style="float:right; border:solid 1px red; width:100px;
height:100px;">p1</p>
<script>
var p = document.getElementsByTagName("p")[0];
```



```
    display(p);                         //切换隐藏
    display(p);                         //切换显示
    display(p);                         //切换隐藏
</script>
```



不管元素是显示或隐藏，如果按如下方式调用，则会显示出来，元素显示为原来的状态：

```
    display(p, true);                   //强制显示
```



视频讲解

15.5.2 设计不透明度

所有现代浏览器都支持元素的透明度，但是不同浏览器对于元素透明度的设置方法不同。IE 浏览器支持 filters 滤镜集，而支持 DOM 标准的浏览器认可 style.opacity 属性。同时，它们设置值的范围也不同，IE 的 opacity 属性值范围为 0～100，其中 0 表示完全透明，而 100 表示不透明；支持 style.opacity 属性浏览器的设置值范围是 0～1，其中 0 表示完全透明，而 1 表示不透明。

【示例 1】为了兼容不同浏览器，可以把设置元素透明度的功能进行函数封装。

```
    //设置元素的透明度
    //参数：e 表示要预设置的元素，n 表示一个数值，取值范围为 0～100，如果省略，则默认为 100，即不透明
显示元素
    //返回值：无
    function setOpacity(e, n){
        var n = parseFloat(n);          //把第二个参数转换为浮点数
        if(n && (n>100) || !n) n=100;
        //如果第二个参数存在且值大于 100，或者不存在该参数，则设置其为 100
        if(n && (n<0))   n =0;          //如果第二个参数存在且值小于 0，则设置其为 0
        if (e.filters){                 //兼容 IE 浏览器
            e.style.filter = "alpha(opacity=" + n + ")";
        }
        else{                           //兼容 DOM 标准
            e.style.opacity = n / 100;
        }
    }
```

在获取元素的透明度时，应注意在 IE 浏览器中不能够直接通过属性读取，而应借助 filters 集合的 item()方法获取 Alpha 对象，然后读取它的 opacity 属性值。

【示例 2】为了避免在读取 IE 浏览器中元素的透明度时发生错误，建议使用 try 语句包含读取语句。

```
    //获取元素的透明度
    //参数：e 表示要预设置的元素
    //返回值：元素的透明度值，范围为 1～100
    function getOpacity(e){
        var r;
        if (! e.filters){
            if (e.style.opacity) return parseFloat(e.style.opacity) * 100;
        }
        try{
            return e.filters.item('alpha').opacity
        }
```



```
        catch(o){
            return 100;
        }
}
```







15.5.3　设计运动对象

运动效果主要通过动态修改元素的坐标来实现，设计的关键有以下两点。

☑ 应考虑元素的初始化坐标、最终坐标，以及移动坐标等定位要素。如果参照物相同，则这个问题比较好解决。

☑ 注意移动的速度、频率等问题。移动可以借助定时器来实现，但效果的模拟涉及算法问题，不同的算法，可能会设计出不同的移动效果，如匀速运动、加速运动和减速运动。在 Flash 动画设计中，专门提供了一个 Tween 类，利用它可以模拟出很多运动效果，如缓动、弹簧震动等效果，其技术核心是算法设计问题。算法好像很高深，如果通俗一点讲，就是通过数学函数计算定时器每次触发时移动的距离。

【示例】下面示例演示如何设计一个简单的元素滑动效果。通过指向元素、移动的位置，以及移动的步数，可以设计按一定的速度把元素从当前位置移动到指定的位置。本示例引用前面介绍的 getB() 方法，该方法能够获取当前元素的绝对定位坐标值。

```
//简单的滑动函数
//参数：e 表示元素，x 和 y 表示要移动的最后坐标位置（相对包含块），t 表示元素移动的步数
function slide(e, x, y, t){
    var t = t || 100; //初始化步数，步数越大，速度越慢，移动的过程越逼真，但是中间移动的误差就越明显
    var o = getB(e);                    //当前元素的绝对定位坐标值
    var x0 = o.x;
    var y0 = o.y;
    var stepx = Math.round((x - x0) / t);
    //计算 x 轴每次移动的步长，由于像素点不可用小数，所以会存在一定的误差
    var stepy = Math.round((y - y0) / t);       //计算 y 轴每次移动的步长
    var out =   setInterval(function(){         //设计定时器
    var o = getB(e);                    //获取每次移动后的绝对定位坐标值
    var x0 = o.x;
    var y0 = o.y;
    e.style["left"] = (x0 + stepx) + 'px';      //定位每次移动的位置
    e.style["top"] = (y0 + stepy) + 'px';       //定位每次移动的位置
    if (Math.abs(x - x0) <= Math.abs(stepx) || Math.abs(y - y0) <=
Math.abs(stepy)) {      //如果距离终点坐标的距离小于步长，则停止循环执行，并校正元素的最终坐标位置
        e.style["left"] = x + 'px';
        e.style["top"] = y + 'px';
        clearTimeout(out);
    };
    }, 2)
};
```

使用时应该定义元素绝对定位或相对定位显示状态，否则移动无效。在网页动画设计中，一般都使用这种定位移动的方式来实现。

```
<style type="text/css">
.block {width:20px; height:20px; position:absolute; left:200px;
```



```
top:200px; background-color:red;}
</style>
<div class="block" id="block1"></div>
<script>
temp1 = document.getElementById('block1');
slide(temp1, 400, 400,60);
</script>
```



Note



视频讲解

15.5.4 设计渐变效果

渐隐渐显效果主要通过动态修改元素的透明度来实现。

【示例】下面示例演示如何实现简单的渐隐渐显动画效果，涉及 setOpacity()函数的调用。

```
//渐隐渐显动画显示函数
//参数：e 表示渐隐渐显元素，t 表示渐隐渐显的速度，值越大，渐隐或渐显的速度越慢，
io 表示渐隐或渐显方式，取值 true 表示渐显，取值 false 表示渐隐
function fade(e, t, io){
    var t = t || 10;                        //初始化渐隐渐显速度
    if(io){                                 //初始化渐隐渐显方式
    var i = 0;
    }else{
    var i = 100;
    }
    var out = setInterval(function(){       //设计定时器
    setOpacity(e, i);                       //调用 setOpacity()函数
    if(io) {                                //根据渐隐或渐显方式决定执行效果
        i ++ ;
        if(i >= 100)    clearTimeout(out);
    }
    else{
        i-- ;
        if(i <= 0)    clearTimeout(out);
    }
    }, t);
}
```

下面调用该函数：

```
<style type="text/css">
.block {width:200px; height:200px; background-color:red;}
</style>
<div class="block" id="block1"></div>
<script>
e = document.getElementById('block1');
fade(e,50,true);                            //应用渐隐渐显动画效果
</script>
```



视频讲解

15.5.5 设计折叠面板

折叠面板在网页中的应用范围比较广，从技术角度分析，它主要利用 CSS 隐藏和显示属性，借助 JavaScript 脚本进行动态控制。本节设计的折叠面板演示效果如图 15.11 和图 15.12 所示。



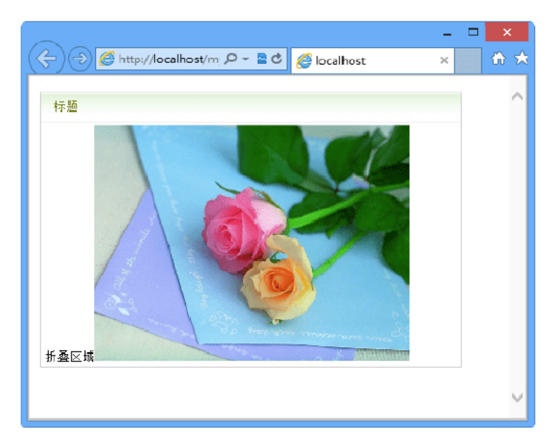


图 15.11　展开面板效果

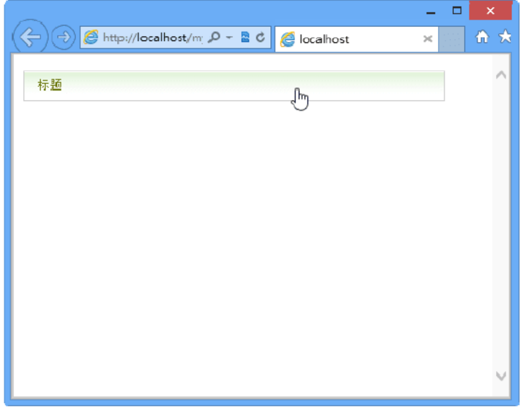


图 15.12　折起面板效果



【操作步骤】

第 1 步，启动 Dreamweaver，新建文档，保存为 test.html。构建 HTML 结构，结构没有特殊要求，使用任何两个标签均可实现折叠效果，但是从语义角度来考虑，使用定义列表是最佳语义结构选择。dl 元素负责构建折叠面板的外框，dt 元素负责构建折叠面板的标题栏，而 dd 元素负责构建面板主体包含框。

```
<dl>
    <dt>标题</dt>
    <dd>折叠区域<img src="images/3.jpg" width="300" /></dd>
</dl>
```

第 2 步，在设计折叠行为之前，先假设这是一个普通的列表结构，然后使用 CSS 来控制定义列表框的表现效果。

```
<style type="text/css">
dl {/*定义列表框样式*/
    width:400px;                                        /*定义折叠面板的宽度，可自定义*/
    border:solid 1px #ccc;                              /*边框，可自定义*/
    font-size:12px;                                     /*字体大小，可自定义*/
}
dt {/*列表标题样式*/
    background:#7FECAD url(images/green.gif) repeat-x;  /*用背景图定义渐变标题背景*/
    color:#71790C;                                      /*字体颜色，可自定义*/
    height:28px;                                        /*标题高度*/
    line-height:28px;                                   /*行高，间接实现垂直对齐*/
    padding-left:1em;                                   /*增加左侧空隙*/
    border-bottom:solid 1px #efefef;                    /*底部边框样式*/
    cursor:pointer;                                     /*鼠标指针为手形*/
}
dd {/*列表项样式*/
    padding:2px 4px;                                    /*增加内容框内边距*/
    margin:0;                                           /*清除缩进*/
}
</style>
```



第 3 步，在上面样式表的基础上定义两个类样式，分别用来隐藏和显示对象。

```
.expand {overflow:visible;}                    /*展开面板时，显示所有内容区域*/
.collapse {                                    /*折叠面板时，仅显示标题区域*/
    height:28px;                               /*限制列表包含框高度，使其等于标题栏高度*/
    overflow:hidden;                           /*强制隐藏多出的区域*/
}
```



Note

第 4 步，完成结构层和表现层的设计后，下面设计交互层。在 JavaScript 脚本中定义一个函数 Switch()用作展开和折叠的开关。

```
<script>
function Switch(dt){ //折叠控制函数
    var dl = dt.parentNode;                                     //获取标题栏的父包含框
    if(dl.className == "collapse")dl.className = "expand";      //如果为折叠，则展开类样式
    else dl.className = "collapse";                             //相反调用折叠类样式
}
</script>
```

第 5 步，完成脚本函数的设计，然后把它绑定到标题栏的 onclick 事件属性上面，代码如下：

```
<dt onclick="Switch(this)">标题</dt>
```

这里使用 this 关键字作为参数进行传递，它代表当前 dt 元素的引用。至此，整个折叠面板的设计就完成了。



视频讲解

15.5.6 设计工具提示

Tooltip（工具提示）是一种比较实用的 JavaScript 应用。当为一个元素（一般为超链接 a 元素）定义 title 属性时，会在鼠标经过时显示提示信息，这些提示能够详细描绘经过对象的包含信息，这对于超链接（特别是图像式超链接）非常有用。同时，搜索引擎也喜欢检索这些信息。

设计思路：使用 DOM 技术获取 title（或其他属性）中的提示信息，然后把这些属性删除，再利用 JavaScript 脚本动态生成一个浮动的层，在层中显示这些提示信息，最后利用 Even 事件对象的鼠标指针坐标属性进行定位。如果结合 CSS 技术，可以把这些浮动的层设计成不同样式，以此达到个性化设计的要求。

【示例 1】本示例不涉及结构层和表现层的设计，为了化繁为简，这里先就一个简单的案例来探索 Tooltip 脚本的实现过程。

【操作步骤】

第 1 步，启动 Dreamweaver，新建文档，保存为 test.html。在文档中设计如下超链接，其提示信息设置为“title="提示信息"”。下面尝试把这个提示信息提取出来，然后删除该属性，最后使用一个新创建的 div 元素动态显示它的位置，并借助 CSS 美化一下该 div 元素。

```
<a href="#" title="提示信息" target="_blank">超链接文本</a>
```

第 2 步，在脚本中获取超链接元素 a，以及该标签设置的 title 属性值。代码如下：

```
var a = document.getElementsByTagName("a")[0];          //获取 a 元素的引用
var tit  = a.getAttribute("title");                     //获取 title 属性值，并存储到一个变量中
```

第 3 步，获取 title 属性值之后，删除该属性，避免它干扰设计。

```
a.removeAttribute("title");                             //移除 title 属性
```





第 4 步，创建一个 div 元素和一个文本节点，把 title 属性值赋给文本节点，然后把文本节点增加到 div 元素内，设置 div 元素样式为绝对定位，并增加一个 class 属性和值，以便于在 CSS 样式表中对该 div 元素进行更个性的控制。

```
var div = document.createElement("div");              //创建 div 元素节点
var   txt = document.createTextNode(tit);              //创建文本节点，并显示 title 属性值
div.style.position = "absolute";                       //为 div 元素定义一个绝对定位
div.setAttribute("class", "title");                    //为 div 元素增加一个类样式，兼容非 IE
div.setAttribute("className", "title");                //为 div 元素增加一个类样式，兼容 IE
div.appendChild(txt);                                  //获取 title 属性值，并传递给 div 元素
```

第 5 步，为超链接 a 元素绑定鼠标经过和鼠标移出的事件处理函数。设计当鼠标指针移过超链接文本时，把创建的 div 元素节点增加到该 a 元素中，而当鼠标指针移出超链接文本时，把 a 元素中增加的 div 节点删除。

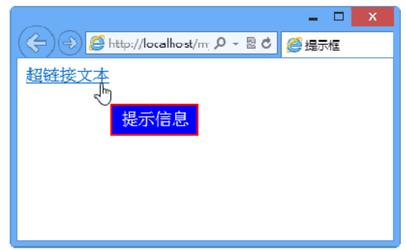
```
a.onmouseover = function(){                            //鼠标经过事件处理函数
    a.appendChild(div);                                //把 div 元素增加到 a 元素中
}
a.onmouseout = function(){                             //鼠标移出事件处理函数
    a.removeChild(div);                                //把 a 元素中的 div 元素删除
}
```

第 6 步，定义鼠标移动事件处理函数，实时跟踪鼠标指针的坐标，并利用该坐标来定位创建的 div 元素的显示位置，以实现它始终显示在鼠标指针的右下角。

第 7 步，在 CSS 样式表中为 title 类定义类样式。

至此，整个提示框效果就设计完成了。在浏览器中预览，则显示效果如图 15.13 和图 15.14 所示。

```
<style type="text/css">
.title {/*提示框类样式*/
    padding:4px 8px;                                   /*增加内侧补白*/
    border:solid 2px red;                              /*设计边框样式*/
    background:blue;                                   /*定义背景为蓝色*/
    color:#fff;                                        /*定义字体为白色*/
    text-decoration:none;                              /*清除受 a 元素影响而产生的下画线*/
}
</style>
```



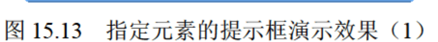


图 15.13　指定元素的提示框演示效果（1）

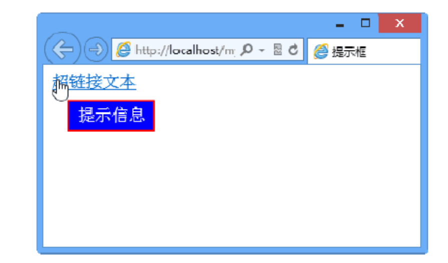


图 15.14　指定元素的提示框演示效果（2）

【示例 2】示例 1 仅就页面中某个具体的超链接来定义提示框，但是在实际设计中无法预测页面中到底有多少超链接，为此需要使用遍历 a 元素节点集合技术来实现动态为页面中所有超链接设计提示框，演示效果如图 15.15 和图 15.16 所示。



Note

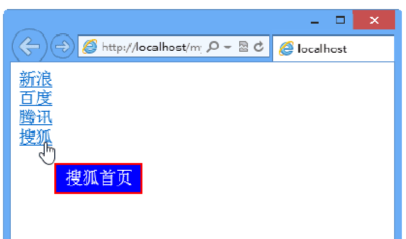


图 15.15　为页面中所有超链接设计提示框演示效果（1）

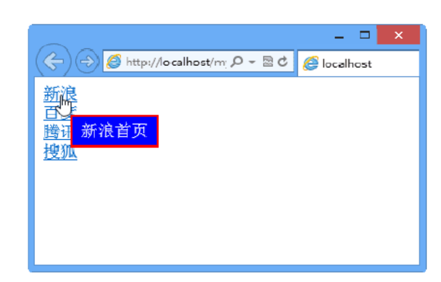


图 15.16　为页面中所有超链接设计提示框演示效果（2）

为实现上述演示效果，本示例依然采用示例 1 的设计思路，只不过中间增加了 for 循环结构来遍历文档中所有的超链接元素 a。

【操作步骤】

第 1 步，复制 test.html，另存为 test2.html。在页面初始化事件中绑定一个事件处理函数，然后获取页面内所有 a 元素的引用。

```
window.onload = function(){
    var a = document.getElementsByTagName("a");            //创建 div 元素节点
}
```

第 2 步，在该事件处理函数内，完成示例 1 的操作。

```
for(var i = 0; i < a.length; i ++ ){                       //遍历页面内所有 a 元素
    tit = a[i].getAttribute("title");                      //获取 a 元素的 title 属性值
    if(tit)   a[i].removeAttribute("title");               //如果属性值存在，则删除该属性
    var div = document.createElement("div");               //创建 div 元素节点
    var txt = document.createTextNode(tit);                //创建并把提示信息赋给文本节点
    div.setAttribute("class", "title");                    //为 div 元素增加类属性，兼容 Firefox
    div.setAttribute("className", "title");                //为 div 元素增加类属性，兼容 IE
    div.style.position = "absolute";                       //绝对定位 div 元素
    div.appendChild(txt);                                  //把文本节点增加到 div 元素
}
```

第 3 步，设计鼠标经过和移出的事件处理函数，以实现增加和删除 div 到 a 元素。考虑到在函数体内定义闭包是无法与外界进行数据交流的，为此在这里主动为闭包函数传递外部动态参数。

```
a[i].onmouseover = (function(i,div){                       //鼠标经过时的事件处理函数
    return function(){                                     //返回处理函数
        a[i].appendChild(div);                             //为 a 元素增加 div 元素
    }
})(i,div);                                 //为闭包函数传递参数，i 表示循环变量值，div 表示引用
a[i].onmouseout = (function(i,div){                        //鼠标移出时的事件处理函数
    return function(){                                     //返回处理函数
        a[i].removeChild(div);                             //为 a 元素移除 div 元素
    }
})(i,div);
```

第 4 步，设计鼠标指针移动的事件处理函数。为了能够兼容不同主流浏览器，以及考虑浏览器窗口可能会出现滚动条，这里使用多个条件结构进行判断来设置指针的坐标值，读者需要了解 Event 对象的属性，详细内容可以参阅第 14 章讲解。



```
    a[i].onmousemove = (function(div,e){//第 1 个参数表示定位元素，第 2 个参数表示事件参数
        return function(e) {//闭包内返回函数体
            var posx = 0, posy = 0;                    //定义两个局部变量，用来存储鼠标指针的坐标
            //判断当前浏览器，如果为 IE，则使用 window.event 获取鼠标指针
            if(e == null) e = window.event;
            //判断是否支持 pageX 或 pageY 事件属性，如果支持，表示浏览器支持 DOM 2.0（如 Firefox 等），
此时可以使用这两个属性获取鼠标指针在窗口中的坐标
            if(e.pageX || e.pageY){
                posx = e.pageX;
                posy = e.pageY;
            }
            else if(e.clientX || e.clientY){//如果不支持 pageX 或 pageY，则使用 clientX 或 clientY
                //如果支持 document.documentElement.scrollTop 属性，则计算指针坐标
                if(document.documentElement.scrollTop){
                    posx = e.clientX + document.documentElement.scrollLeft;
                    posy = e.clientY + document.documentElement.scrollTop;
                }else{//否则使用传统的方法来计算指针的坐标位置
                    posx = e.clientX + document.body.scrollLeft;
                    posy = e.clientY + document.body.scrollTop;
                }
            }
            div.style.top = (posy + 20) + "px";        //把鼠标指针的 y 坐标作为定位值赋给 div
            div.style.left = (posx + 10) + "px";       //把鼠标指针的 x 坐标作为定位值赋给 div
        }
    })(div);
```

提示： documentElement 在 DOM 2.0 中表示 html 元素，因此要获取当前页面的滚动条纵坐标位置，可以使用如下方法：

document.documentElement.scrollTop;

如果浏览器不支持 documentElement 对象，则可以使用如下方法获取滚动条纵坐标位置：

document.body.scrollTop;

这里 body 对象表示 body 元素，而 document 表示文档对象，scrollTop 属性表示滚动条的纵坐标。在标准 W3C 下，document.body.scrollTop 始终为 0，需要使用 document.documentElement.scrollTop 来获取滚动条坐标。IE 浏览器从 5.5 版本开始不再支持 document.body.scrollTop 和 document.body. scrollLeft 方法，所以在编程中一般使用示例中的方法来进行判断。

15.6　在 线 练 习

本节专题练习使用 JavaScript 控制 CSS 脚本样式，感兴趣的读者可以扫码练习。



在 线 练 习



使用 CSS 设计 XML 文档样式

XML 是一种允许用户自定义标签的标识语言，用法比 HTML 灵活，功能更强大。XML 可以标识数据，作为一种通用数据格式，非常适合 Web 传输，XML 也是传统桌面应用中比较流行的数据存储和交换格式。本章将讲解如何使用 CSS 设计 XML 文档样式。

【学习重点】

⏭ 熟悉 XML 文档结构。

⏭ 能够使用 CSS 设计 XML 显示样式。



16.1　XML 样式基础





XML 是可扩展标签语言的英文首字母缩写，在结构上与 HTML 相似，但是 XML 优势明显：通用、结构简洁，非常适合各种网络应用的需要。

16.1.1　XML 文档结构

XML 文档也是文本文件，扩展名为.xml。XML 文档结构一般包含 3 部分：XML 声明、处理指令和 XML 元素。其中处理指令是可选部分。

【示例】新建文本文件，保存为 test.xml，然后输入下面代码。

```
<?xml version="1.0" encoding="utf-8"?>
<blog>
    <item>
    <id>1</id>
        <title>标题</title>
        <time>发布时间</time>
        <content>日志内容</content>
        <word>
            <user>昵称</user>
            <time>留言时间</time>
            <text>留言内容</text>
        </word>
    </item>
</blog>
```

上面代码是一个非常简单的 XML 文档，与 HTML 文档结构相似，但是标签可以随意命名，没有任何默认格式，仅是数据信息的语义标识。在浏览器中预览，显示效果如图 16.1 所示。

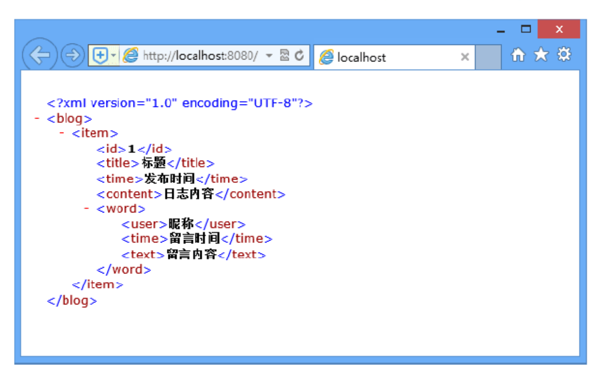


图 16.1　XML 文档显示效果

在上面的代码中，第 1 行代码表示 XML 声明，从第 2 行开始是各种标签嵌套在一起。与 HTML 一样，XML 也是一个基于文本的标签语言，不同的是，XML 标签说明了数据的含义，而不是如何显示它。

【拓展】

XML 文档内容都是由一个根节点构成的（如 blog），它由开始标签<blog>和结束标签</blog>组成。





Note

开始标签与结束标签之间就是这个元素的内容。由于各个元素的内容被各自的元素标签所包含，在XML 中各种数据的分类查找和处理变得非常容易。

XML 标签包含 3 部分：标签的起始符“<”、标签的名称和标签的终止符“>”，标签的名称也称为元素，它表示一个对象。标签的起始和终止符分别为 ASCII 编码的小于号和大于号。在 HTML 中标签都是预定义的，包含默认的显示样式，而在 XML 中标签名称是不固定的，可以根据需要来定义和使用标签。XML 也支持空标签，如<blog></blog>，一般可以简写为< blog />。

在 XML 中可以根据需要为标签定义属性。在开始标签或空标签中可以包含多个属性，属性的作用是对标签及其内容的附加信息进行描述。

XML 属性由使用空格分隔开的名/值对构成。所有的属性值都必须使用引号括起来。语法形式如下：

```
<标签名 属性名 1="属性值 1" 属性名 2="属性值 2" 属性名 3="属性值 3"...>元素内容</标签名>
```

例如：

```
<留言 姓名="张三" QQ="666666666" Email="zhangsan@263.net" 留言时间="2017-4-5 16:39:26">
        这里是我的留言
</留言>
```

对于空元素，其语法形式如下：

```
<标签名 属性名 1="属性值 1" 属性名 2="属性值 2" 属性名 3="属性值 3"... / >
```

元素和属性都可以描述信息，那么该使用属性，还是使用元素呢？一般来讲，具有如下特征的信息可以考虑使用属性来表示。

☑ 与文档无关的简单信息。例如，<书桌 长="240cm" 宽="80cm" 高="100cm" />中的“书桌”元素，其目的是向用户展示一个书桌，但书桌的大小与用户基本无关，而且其“长、宽和高”也没有子结构。在这种情况下，就可以将矩形的“长、宽和高”信息作为元素的属性进行定义。

☑ 与文档有关，而与文档的内容无关的简单信息。例如，<Email 发送时间="2017-4-5 16:39:26" 发送人="张三" >这里是电子邮件内容</Email>。

当然，有很多信息既可以用元素来表示，也可以用属性来表示。例如，对于上面示例中的留言信息，以及与留言相关的属性，这些留言属性既可以使用元素来表示，也可以使用属性来表示。

提示：在将已有文档处理为 XML 文档时，文档的原始内容应全部表示为元素；而编写者所增加的一些附加信息，如对文档某一点内容的说明、注释、文档的某些背景材料等信息可以表示为属性，当然前提是这些信息非常简单。

在创建和编写 XML 文档时，希望显示的内容应表示为元素，即能够在浏览器中显示出来的信息，反之表示为属性。

实在无法确定表示为元素或属性的，就可以表示为元素。因为对于文档处理来讲，元素比属性容易操作。



视频讲解

16.1.2 嵌入 CSS 样式

在 XML 中应用 CSS 的方式有 3 种，这与 HTML 应用 CSS 相似。

☑ 内部声明 CSS：在 XML 文档中直接置入 CSS 样式。

☑ 连接 CSS 样式表：新建 CSS 样式表文件，然后使用命令将 XML 文档与 CSS 文件关联在一起。

☑ 内置 CSS 样式表：新建 CSS 样式表文件，然后在 XML 文档内部样式表中使用@import 导入。



XML 文档内部包含的 CSS 样式被称为内部 CSS。为了将 CSS 样式置入 XML 文档内部，并让处理器识别哪些是 CSS 样式，哪些是 XML 元素，需要引入<style>标签，并且通过名域机制引入<style>，具体格式如下：

```
<根元素名 xmlns:html="http://www.w3.org/TR/REC-HTML40">
```



Note

这是一个名域声明语句，http://www.w3.org/TR/REC-HTML40 是定义 HTML 标签的名域（文件），为了在 XML 文档中引入 HTML 的标签<style>和</style>，必须指定它的来源和作用，然后把所有的在 XML 文档中可能出现的 CSS 样式指令都放在<style>标签对中。HTML 标签的名域文件有很多，根据文档类型可以酌情设置。例如，下面声明定义 XHTML1 文档类型。

```
<根元素名 xmlns:html=" http://www.w3.org/1999/xhtml ">
```

注意：有了这条名域声明语句后，在 XML 文档中使用 “HTML:” 为前缀，后面可指定任何 HTML 标签，如<HTML:DIV>和</HTML:DIV>、<HTML:A>和</HTML:A>等。

具有内部 CSS 的 XML 文档引用方式如下：

```
<html:style>。
```

这里引入的 HTML 标签来自 HTML4 文档类型。格式如下：

```
<?xml version="1.0" encoding="gb2312" standalone="yes" ?>
<?xml-stylesheet type="text/css" ?>
<根元素 xmlns:html="http://www.w3.org/TR/REC-HTML40">
    <html:style>
        CSS-selector1{属性名:属性值; 属性名:属性值; ……}
        CSS-selector2{属性名:属性值; 属性名:属性值; ……}
        …
    </html:style>
    <XML 元素 1……>元素内容</XML 元素 1>
    <XML 元素 1……>元素内容</XML 元素 1>
    …
</根元素>
```

【示例】在下面示例中，通过在 XML 文档内部定义样式表，实现对 XML 文档显示样式进行控制，显示效果如图 16.2 所示。注意 HTML 名域 URL 的引用。

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/css"?>
<poetry xmlns:html="http://www.w3.org/1999/xhtml">
    <html:style>
    poetry {
        background-color: #FFC;
        display: block;
        margin: 2em;
    }
    head {
        display: block;
        font-size: 32px;
        color: red;
        text-align: center;
```



Note

```
        }
        author {
            display: block;
            font-size: 20px;
            color: blue;
            text-align: center;
            margin:12px 0;
        }
        content {
            display: block;
            font-size: 16px;
            color: green;
            text-align: center;
            line-height:1.8em;
            padding-left: 2em;
        }
        </html:style>
        <head>静夜思</head>
        <author>李白</author>
        <content>床前明月光，疑是地上霜。</content>
        <content>举头望明月，低头思故乡。</content>
    </poetry>
```



图 16.2　通过内部 CSS 样式控制 XML 文档效果

在上面的代码中，第 1 行是 XML 文件的头部声明，作为一个格式良好的 XML 文档，应该添加头部的声明信息。第 2 行是 CSS 样式的声明，其中，xml-stylesheet 的意思是为 XML 文档添加样式表，type="text/css"的意思是样式表的类型是 CSS 样式表。第 4 行和第 31 行中间的内容则是 CSS 样式的内容，其中第 4 行用来声明添加 CSS 代码，第 31 行则是它的封闭标签。后面的几行是 XML 文档的内容。在添加了 CSS 样式之后，在 IE 中打开这个 XML 文档，效果如图 16.2 所示。



视频讲解

16.1.3　使用 CSS 样式表

一般建议使用外部 CSS 样式表，将所有规范 XML 元素的样式用一个独立的 CSS 文件保存，使用起来会更方便。

建立 CSS 样式表文件后，需要将 CSS 文档与对应的 XML 文档关联，这是在 XML 文档头部用语句来完成的：

```
<?xml-stylesheet type="text/css" href="CSS 文档的 URL" ?>
```

在这条声明语句中，type 用来指定文档类型，这里表明是文本、CSS 类型。href 用来指明引用的





CSS 文档的位置、名称和扩展名，即 CSS 文档的通用资源定位地址。

【示例】下面示例演示如何导入外部样式表文件。

【操作步骤】

第 1 步，新建 XML 文档，保存为 index.xml。

第 2 步，输入下面代码，设计与唐诗相关的标识内容。

```
<?xml version="1.0" encoding="utf-8"?>
<poetry>
    <head>鹿柴</head>
    <author>王维</author>
    <content>空山不见人，但闻人语响。</content>
    <content>返影入深林，复照青苔上。</content>
</poetry>
```

第 3 步，新建 CSS 样式表文件，保存为 style.css。

第 4 步，输入下面样式代码。

```
poetry {
    background-color: #FFC;
    display: block;
    margin: 2em;
}
head {
    display: block;
    font-size: 32px;
    color: red;
    text-align: center;
}
author {
    display: block;
    font-size: 20px;
    color: blue;
    text-align: center;
    margin: 12px 0;
}
content {
    display: block;
    font-size: 16px;
    color: green;
    text-align: center;
    line-height: 1.8em;
    padding-left: 2em;
    text-shadow:2px 2px 2px #93FB40;
}
```

第 5 步，在 index.xml 文档头部位置，即第 2 行位置，使用<?xml-stylesheet>命令导入外部样式表文件 style.css。

```
<?xml-stylesheet type="text/css" href="style.css"?>
```

第 6 步，保存文档，在浏览器中预览，则显示效果如图 16.3 所示。





如果没有 CSS 样式表的作用，在浏览器中预览 index.xml 文档，则显示效果如图 16.4 所示。

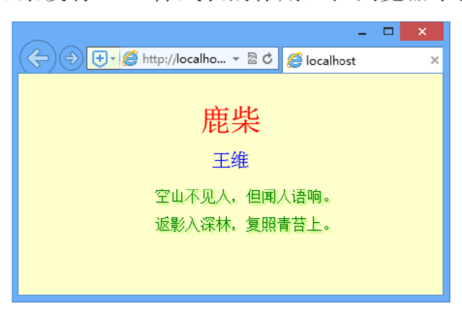


图 16.3　通过外部 CSS 样式表控制 XML 文档效果

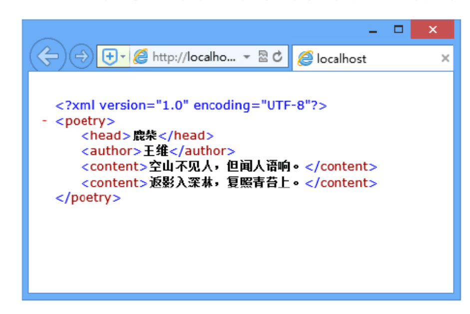


图 16.4　无 CSS 的 XML 文档效果

16.2　案例实战

本节将通过案例的形式帮助读者使用 CSS 设计 XML 文档样式，以提高实战技法和技巧，快速理解 CSS 在 XML 中的应用。


视频讲解

16.2.1　设计特效文字

本节案例介绍如何使用 CSS 定位技术设计一个文字阴影特效，如图 16.5 所示。



图 16.5　设计 XML 文字特效

【操作步骤】

第 1 步，新建 XML 文档，保存为 index.xml。

第 2 步，与使用 HTML 制作文字阴影效果的思路基本一致，用两个标记分别记录两段相同的文字，XML 文档代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<shadow>
    <char1>春眠不觉晓，处处闻啼鸟。</char1>
    <char2>春眠不觉晓，处处闻啼鸟。</char2>
</shadow>
```

第 3 步，新建 CSS 样式表文件，保存为 style.css。

第 4 步，输入下面样式代码。






```
shadow {
    font-family: Arial;
    font-size: 80px;
    font-weight: bold;
}
char1 {
    position: absolute;         /*绝对定位*/
    color: #FFFF00;
    top: 10px;
    left: 15px;
    z-index: 2;                 /*高低关系*/
    border: 2px solid #222;
    padding: 5px 10px 5px 10px;
}
char2 {
    position: absolute;         /*绝对定位*/
    top: 15px;
    left: 20px;
    color: #ff0000;
    z-index: 1;                 /*高低关系*/
    padding: 5px 10px 5px 10px;
    background-color: #7c0000;
}
```

第 5 步，在 index.xml 文档头部位置，即第 2 行位置，使用<?xml-stylesheet>命令导入外部样式表文件 style.css，即可完成本例效果。

```
<?xml-stylesheet type="text/css" href="style.css"?>
```

16.2.2　设计表格样式


视频讲解

本节案例介绍如何使用 CSS 设计 XML 以表格样式显示数据，并隔行换色，效果如图 16.6 所示。



表格标签

标签	描述
<table>	定义表格。
<caption>	定义表格标题。
<th>	定义表格中的表头单元格。
<tr>	定义表格中的行。
<td>	定义表格中的单元。
<thead>	定义表格中的表头内容。
<tbody>	定义表格中的主体内容。
<tfoot>	定义表格中的表注内容（脚注）。
<col>	定义表格中一个或多个列的属性值。
<colgroup>	定义表格中供格式化的列组。

图 16.6　设计表格显示样式

【操作步骤】

第 1 步，新建 XML 文档，保存为 index.xml。





第 2 步，设计 XML 文档结构。使用<list>定义表格框，使用<caption>定义表头，使用<title>定义列标题行，使用<item>定义数据行，使用<name>定义第 1 列单元格，使用<describe>定义第 2 列单元格。

```
<?xml version="1.0" encoding="gb2312"?>
<list>
    <caption>表格标签</caption>
    <title>
        <name>标签</name>
        <describe>描述</describe>
    </title>
    <item>
        <name>&lt;table&gt;</name>
        <describe>定义表格。</describe>
    </item>
    <item>
        <name>&lt;caption&gt;</name>
        <describe>定义表格标题。</describe>
    </item>
    <item>
        <name>&lt;th&gt;</name>
        <describe>定义表格中的表头单元格。</describe>
    </item>
    <item>
        <name>&lt;tr&gt;</name>
        <describe>定义表格中的行。</describe>
    </item>
    <item>
        <name>&lt;td&gt;</name>
        <describe>定义表格中的单元。</describe>
    </item>
    <item>
        <name>&lt;thead&gt;</name>
        <describe>定义表格中的表头内容。</describe>
    </item>
    <item>
        <name>&lt;tbody&gt;</name>
        <describe>定义表格中的主体内容。</describe>
    </item>
    <item>
        <name>&lt;tfoot&gt;</name>
        <describe>定义表格中的表注内容（脚注）。</describe>
    </item>
    <item>
        <name>&lt;col&gt;</name>
        <describe>定义表格中一个或多个列的属性值。</describe>
    </item>
    <item>
        <name>&lt;colgroup&gt;</name>
        <describe>定义表格中供格式化的列组。</describe>
    </item>
</list>
```



第 3 步，新建 CSS 样式表文件，保存为 style.css。

第 4 步，输入下面样式代码。

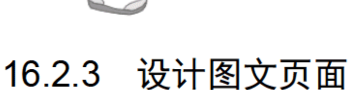
```
list {
    font-family: Arial;
    font-size: 14px;
    position: absolute;          /*绝对定位*/
    top: 0px;
    left: 0px;
    padding: 4px;                /*适当地调整位置*/
}
caption {
    margin-bottom: 6px;
    font-weight: bold;
    font-size: 1.4em;
    text-align:center;
    display: block;              /*块元素*/
}
title {
    background-color: #4bacff;
    display: block;              /*块元素*/
    border: 1px solid #0058a3;   /*边框*/
    margin-bottom: -1px;         /*解决边框重叠的问题*/
    padding: 4px 0px 4px 0px;
    font-weight:bold;
}
item {
    display: block;              /*块元素*/
    background-color: #eaf5ff;   /*背景色*/
    border: 1px solid #0058a3;   /*边框*/
    margin-bottom: -1px;         /*解决边框重叠的问题*/
    padding: 4px 0px 4px 0px;    /*Firefox 不支持行内元素的 padding，只支持 block 元素的 padding，为了尽
量统一两个浏览器，将 padding-top 和 bottom 放到这里设置*/
}
item:nth-child(2n+1) {
    background-color: #eee;      /*背景色*/
}
name, describe {
    padding: 2px 8px 2px 8px;
    display: inline-block;
}
name {                           /*Firefox 不支持行内元素的 width 属性*/
    width: 100px;}
describe {width: 300px;}
```

第 5 步，在 index.xml 文档头部位置，即第 2 行位置，使用<?xml-stylesheet>命令导入外部样式表文件 style.css，即可完成本例效果。

```
<?xml-stylesheet type="text/css" href="style.css"?>
```


视频讲解


Note



16.2.3 设计图文页面

本案例设计一个图文并茂的诗配画效果，在页面中通过夕阳背景衬托意境，通过 CSS 嵌入的诗人速写给画面以点睛，并在背景图的左上角显示一首唐代诗人李商隐的《登乐游原》小诗，使页面看起来更富诗情画意，设计效果如图 16.7 所示。



图 16.7　设计诗情画意图文效果

【操作步骤】

第 1 步，构建 XML 文档结构，并保存为 index.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<poem>
    <title>登乐游原</title>
    <author>唐  李商隐</author>
    <wen>
        <li>向晚意不适，</li>
        <li>驱车登古原。</li>
        <li>夕阳无限好，</li>
        <li>只是近黄昏。</li>
    </wen>
</poem>
```

第 2 步，新建 CSS 样式表文件，保存为 xml.css。在样式表文件中定义 XML 文档中各个标签的基本显示样式。

```
poem { /*画面样式*/
    margin:0px;
    background-image:url(06.jpg);   /*设计画面背景图*/
}
title { /*标题样式*/
    position:absolute;
    left:80px;
    top:20px;
    font-size:26px;
```








```
        color:#FFF;
        font-weight:bold;
    }
    author { /*作者样式*/
        position:absolute;
        left:100px;
        top:60px;
        font-size:14px;
        color:#0033FF;
    }
    wen { /*诗文外包含框样式*/
        position:absolute;
        left:80px;
        top:90px;
    }
    li { /*诗文列表样式*/
        display:block;
        color:#000;
        font-size:20px;
        font-weight:bold;
        margin:6px;
    }
```

第 3 步，使用 CSS 在文档中嵌入诗人画像，并通过 width 和 height 属性定义诗文外包含框的大小，并用 background 属性定位值 bottom 和 right 把嵌入的诗人画像定位到包含框的右下角。

```
    wen {
        width:620px;
        height:350px;
        background:url(author.png) bottom right no-repeat ;
    }
```

16.2.4　设计正文版面



图文混排多用于正文内容部分或者新闻内容部分，处理的方式也很简单，文字一般围绕在图片的一侧，或者四周。这样的设计可以让整个版面显得饱满，又不杂乱。为了获取较高的代码可移植性，要求使用 XML+CSS 方式来实现。

图文混排版式一般情况下不是在页面设计过程中实现的，而是在后期通过网站的新闻发布系统进行自动发布，这样的内容发布模式对于图片的大小、段落文本排版都属于不可控的范围，因此要考虑图与文不规则问题。

使用绝对定位方式后，图片将脱离文档流，成为页面中具有层叠效果的一个元素，将会覆盖文字，因此不建议使用绝对定位实现图文混排。通过浮动设计图文混排是比较理想的方式，适当利用补白（padding）或者文字缩进（text-indent）的方式将被图片与文字分开。本例设计效果如图 16.8 所示。

【操作步骤】

第 1 步，构建 XML 文档结构，并保存为 index.xml。

```
<?xml version="1.0" encoding="gb2312"?>
<?xml-stylesheet type="text/css" href="images/xml.css"?>
<new>
```

```
        <h1>月活 4 亿、“市值仅次于腾讯”、“保值力压苹果”，逆天的美图却遭变现困难</h1>
        <detail>
            <time>发布时间：2016.10.00 00:00</time>
            <from>来源：Eastland</from>
        </detail>
        <pic>
            <img></img>
            <title>美图人物</title>
        </pic>
        <p>今年 8 月 22 日，美图招股文件在香港交易所官网亮相。文件披露：2016 年 6 月美图旗下 App 的月活用户达到 4.46 亿；在主流社交网站分享的照片中，53.5%经过美图处理。</p>
        <p>随着美图挂牌日期的临近，一些“有意思”的信息悄然流传。</p>
        <p>比如，美图市值将超过 400 亿港元，成为“香港主板市值仅次于腾讯的互联网企业”。截至 2016 年 9 月底，香港主板 1930 家公司总市值 25.6 万亿港元，其中腾讯市值超过 2 万亿港元，占整个主板市值的 7.9%。</p>
        <p>还有一则新闻说美图 M6 上市 4 个月后“仍是一机难求，溢价幅度赶 iPhone7”、“最保值手机不是苹果、三星而是美图手机！”</p>
        <p>市值拿腾讯说事儿，“畅销、保值”用苹果、三星“垫背”，美图的标杆（benchmark）令人敬畏。</p>
        <p>美图很好用，但研发出好应用的公司效益不一定就好，4.5 亿用户、市值仅次于腾讯、保值力压苹果终究不过是“噱头”。</p>
    </new>
```



图 16.8　设计正文版面效果

整个结构包含在<new>新闻框中，新闻框中包含 4 部分，第 1 部分是新闻标题，由<h1>标题标签负责；第 2 部分是新闻的附加信息，由<detail>标签负责管理，包括发布时间标签<time>和新闻源自标签<from>；第 3 部分是新闻图片，由<pic>图片框负责控制，其中包含<img>标签，负责显示图片，<title>标签，负责注释图片；第 4 部分是新闻正文部分，由<p>标签负责管理。

第 2 步，新建 CSS 样式表文件，保存为 xml.css。在样式表文件中定义 XML 文档中各个标签的基本显示样式。先输入下面样式，定义新闻框显示效果。

```
    new {
        display:block;
        width:900px; /*控制内容区域的宽度，根据实际情况考虑，也可以不需要*/
```



```
        margin:12px;
    }
```

第 3 步，继续添加样式。设计新闻标题样式，其中包括 3 级标题，统一标题为居中显示对齐，一级标题字体大小为 28px，<detail>标签字体大小为 14px，<title>标签字体大小为 12px，同时<title>标签标题取消默认的上下边界样式。



```
    new h1 {
        display:block;
        text-align:center;
        font-size:28px;
        margin:1em;
    }
    new detail {
        display:block;
        text-align:center;
        font-size:14px;
        margin:1em;
    }
    new time, new from {
        padding-right:12px;
        ;
    }
    new title {
        display:block;
        text-align:center;
        font-size:12px;
        margin:0;
        padding:0;
    }
```

第 4 步，设计新闻图片框和图片样式。设计新闻图片框向左浮动，然后定义新闻图片大小固定，并适当拉开与环绕的文字之间的距离。

```
    new pic {
        display:block;
        float:left;
        text-align:center;
    }
    new img {
        display:inline-block;
        background:url(01.jpg);
        width:307px;
        height:409px;
        margin-right:1em;
        margin-bottom:1em;
    }
```

第 5 步，设计段落文本样式，主要包括段落文本的首行缩进和行高效果。

```
    new p {
        display:block;
```



```
        line-height:1.8em;
        text-indent:2em;
    }
```



简单的几句 CSS 样式代码就能实现图文混排的页面效果。其中重点内容是将图片设置为浮动，float:left 就是将图片向左浮动，那么，如果设置为 float:right 将会是什么效果呢？读者可以修改代码并在浏览器中查看页面效果。

16.3 在线练习

本节专题练习文档对象模型开发的一般方法，感兴趣的读者可以扫码练习。



在线练习



综合实战：设计响应式网站

随着各种智能设备的推广和普及，网站的建设者需要满足访问者通过智能手机、平板电脑、笔记本电脑、台式机、电视机，以及未来任何可以上网的设备获取信息的需求。响应式Web设计就是为此诞生的。本章将通过一个综合案例讲解如何构建在各种设备上都能正常工作的网站，它能根据设备的功能和特征对布局进行调整。

【学习重点】

- 创建可伸缩图像。
- 创建弹性布局网格。
- 理解和实现媒体查询。



17.1 认识响应式 Web 设计



网站设计主要有两大类型：固定宽度和响应式。

对于固定（fixed）布局，整个页面和每一栏都有基于像素的宽度。顾名思义，无论是使用移动电话和平板电脑等较小的设备查看页面，还是使用桌面浏览器并对窗口进行缩小，它的宽度都不会改变。在引入响应式 Web 设计之前，这是大多数网站选用的布局方式，也是学习 CSS 时最容易掌握的布局方式。

响应式页面也称为流式（fluid 或 liquid）页面，它使用百分数定义宽度，允许页面随显示环境的改变进行放大或缩小。除了具有流动栏，响应式页面还可以根据屏幕尺寸以特定方式调整其设计。例如，可以更改图像大小或者调整每一栏，使其大小更合适。这就可以在使用相同 HTML 的情况下，为移动用户、平板电脑用户和桌面用户定制单独的体验，而不是提供 3 个独立的网站。

没有一种布局方式可以适用于所有的情景。不过，随着智能手机和平板电脑的广泛应用，未来一定还会出现各种不同尺寸的智能设备，因此在设计网页时有必要将网站做成响应式布局。这也是每天都有大量响应式网站出现的原因。

【拓展】

响应式 Web 设计起源于 Ethan Marcotte，他创建了术语“响应式 Web 设计”（Responsive Web Design），并向大家介绍了创建响应式网站的技术。人们首次广泛关注这种方法始于他发表在 *A List Apart* 上的文章（www.alistapart.com/articles/responsive-web-design/）。他在 *Responsive Web Design* 一书中对此做了更为深入细致的探讨。

Ethan Marcotte 的方法包含以下 3 点。

☑ 灵活的图像和媒体。图像和媒体资源的尺寸是用百分数定义的，从而可以根据环境进行缩放。

☑ 灵活的、基于网格的布局，也就是流式布局。对于响应式网站，所有的 width 属性都用百分数设定，因此所有的布局成分都是相对的。其他水平属性通常也会使用相对单位（如 em、百分数和 rem 等）。

☑ 媒体查询。使用这项技术，可以根据媒体特征（如浏览器可视页面区域的宽度）对设计进行调整。

John Allsopp 于 2000 年发表《Web 设计之道》（*A Dao of Web Design*，http://alistapart.com/article/dao），该文讨论了设计和构建灵活的网站的方法。这篇文章是响应式 Web 设计的先驱，Marcotte 以及很多其他作者都引用过这篇文章，影响巨大。Jeremy Keith 在题为 *One Web* 的演讲中归纳了“一个网站适应所有设备”的方法。

【参考】

Screen Sizes 网站（http://screensiz.es）提供了流行设备与显示屏的分辨率和设备宽度信息，使用媒体查询时，这些信息很有用。

Maximiliano Firtman 维护了一个现代移动设备对 HTML5 和 CSS3 支持情况的表格（参考 http://mobilehtml5.org），其中大量信息属于 HTML5 高级特性。

17.2 构 建 页 面

高效网页的核心是结构良好、语义化的 HTML。下面就来具体介绍如何构建本章案例的具体结构。



Note

【操作步骤】

第 1 步，恰当地使用 article、aside、nav、section、header、footer 和 div 等元素将页面划分成不同的逻辑区块。本例创建了一个虚构博客，页面结构要点说明如下：

☑ 使用 3 个 div，其中一个将整个页面包起来，另外两个将两部分主体内容区域包起来，以便应用样式设计。

☑ 用作报头的 header，包括标识、社交媒体网站链接和主导航。

☑ 划分为多个博客条目 section 元素的 main 元素，其中每个 section 元素都有自己的页脚。

☑ 附注栏 div（同时使用了 article 和 aside），提供关于博客作者和右栏（应用 CSS 之后就有了）博客条目的链接。

☑ 页面级 footer 元素，包含版权信息等内容。

第 2 步，按照一定的顺序放置内容，确保页面在不使用 CSS 的情况下也是合理的。例如，首先是报头，接着是主体内容，然后是一个或多个附注栏，最后是页面级的页脚。将最重要的内容放在最上面，对于智能手机和平板电脑等小屏幕用户来说，不用滑动屏幕太远就能获取主体内容。此外，搜索引擎“看到”的页面也类似于未应用 CSS 的页面，因此，如果将主体内容提前，搜索引擎就能更好地对网站进行索引。这同样也会让屏幕阅读器用户受益。

第 3 步，以一致的方式使用标题元素（h1～h6），从而明确地标识页面上这些区块的信息，并对它们按优先级排序。

第 4 步，使用合适的语义标记剩余的内容，如段落、图和列表。

第 5 步，如果有必要，使用注释来标识页面上不同的区域及其内容。根据个人习惯，选用一种不同的注释格式标记区块的开始（而非结束）。

页面基本框架结构代码如下：

```
<div class="page">
    <!-- ==== 开始报头 ==== -->
    <header class="masthead" role="banner">
        <p class="logo"><a href="/"><img /></a></p>
        <ul class="social-sites">[社交图片链接]</ul>
        <nav role="navigation">[主导航链接列表]</nav>
    </header>
    <!--结束报头-->
    <div class="container">
        <!-- ==== 开始主体内容 ==== -->
        <main role="main">
            <section class="post">
                <h1>[文章标题]</h1>
                <img   class="post-photo-full" />
                <div class="post-blurb">
                    <p>[正文内容]</p>
                </div>
                <footer class="footer">[博客条目页脚]</footer>
            </section>
            <section class="post">
                <h1>[文章标题]</h1>
                <img   class="post-photo" />
                <div class="post-blurb">
```





Note

```
                    <p>[正文内容]</p>
                </div>
                <footer class="footer">[博客条目页脚]</footer>
            </section>
            <nav role="navigation">
                <ol class="pagination">[链接列表项]</ol>
            </nav>
        </main>
        <!--结束主体内容-->
        <!-- ==== 开始附注栏 ==== -->
        <div class="sidebar">
            <article class="about">
                <h2>关于自己</h2>
            </article>
            <div class="mod">
                <h2>我的经历</h2>
                ... [映射图像] ... </div>
            <aside class="mod">
                <h2>热门职位</h2>
                <ul class="links">[链接列表项]</ul>
            </aside>
            <aside class="mod">
                <h2>最近分享</h2>
                <ul class="links">[链接列表项]</ul>
            </aside>
        </div>
        <!--结束附注栏-->
    </div>
    <!--结束容器-->
    <!-- ==== 开始页脚 ==== -->
    <footer role="contentinfo" class="footer">
        <p class="legal"><small>[版权信息]</small></p>
    </footer>
    <!--结束页脚-->
</div>
<!--结束页面-->
```

在上面的结构中，使用 section 元素来标记每个包含部分博文的条目。如果它们是完整的博文条目，就应该使用 article 来标记，例如，在单独的、完整的博客文章页内。对它们使用 article 替代 section 也可以，只要代表这一段代码可以成为独立的内容即可。

第 6 步，保存文档，在浏览器中预览，则显示效果如图 17.1 所示。这个页面是一栏的，除了浏览器默认样式以外并没有设置其他样式。由于它的语义结构非常好，因此页面是完全可用且可理解的。

注意： 不一定要在应用 CSS 之前就标记好整个页面。实践中，很少先将一个区块的 HTML 写好，再为其编写一些或全部的 CSS，然后对下一个区块重复这一过程。处理方式取决于个人习惯。





Note



图 17.1　页面结构默认显示效果

17.3　设计基本样式

构建并完善页面结构和内容之后，下面就来重点介绍网页样式的设计。

提示：在学习之前，读者应该有一定的 CSS 基础，没有基础的建议先阅读小白手册，等入门后再接着学习。

17.3.1　兼容早期浏览器

现代浏览器原生支持 HTML5 新增结构元素。从样式的角度来说，这意味着浏览器将为这些新的元素应用默认样式。例如，article、footer、header、nav 以及其他一些元素显示为单独的行，就像 div、blockquote、p 以及其他元素在 HTML5 之前的版本中称作块级元素。

【操作步骤】

第 1 步，大部分浏览器允许对它们并不原生支持的元素添加样式，样式代码如下：

```
article, aside, figcaption, figure, footer, header, main, nav, section {
    display: block;
}
```

大多数浏览器默认将它们无法识别的元素作为行内元素处理。因此，这一小段 CSS 将强制 HTML5 新语义元素显示在单独的行。

提示：如果使用 CSS 重置或 normalize.css，可以跳过这一步，它们会包含这里的代码。

第 2 步，大部分浏览器会忽略它们不原生支持的元素的 CSS。HTML5 shiv 是专门用于解决这一问题的一段 JavaScript。在每个页面的 head 元素（注意不是 header 元素）中添加下面的代码，实现在 IE9 之前的 IE 中为新的 HTML5 元素设置样式。

```
<!--[if lt IE 9]>
    <script src="js/html5shiv.js"></script>
<![endif]-->
```



17.3.2 重置默认样式



Note

每个浏览器都有内置的默认样式表，HTML 会遵照该样式表显示网页。整体上，不同浏览器提供的默认样式表是相似的，但也存在一定的差异。为此，开发人员在应用他们自己的 CSS 之前，常常需要抹平这些差异。抹平差异的方法主要有两种。

☑ 使用 CSS 重置（reset）样式表，如 Eric Meyer 创建的 Meyer 重置（http://meyerweb.com/eric/tools/css/reset/）。另外还有其他的一些重置样式表。

☑ 使用 Nicolas Gallagher 和 Jonathan Neal 创建的 normalize.css 开始主样式表。该样式表位于 http://necolas.github.com/normalize.css/。

CSS 重置可以有效地将所有默认样式都设为“零”。第二种方法，即使用 normalize.css，则采取了不同的方式。它并非对所有样式进行重置，而是对默认样式进行微调，这样确保在不同的浏览器中具有相似的外观。

用户也可以保留浏览器的默认样式，并自己编写相应的 CSS。如果确实要使用 normalize.css 或 CSS 重置，也不必保留它们提供的所有 CSS。本例使用了 normalize.css 中的一部分代码，并对文本添加了一些样式，形成一个初始的页面。

```
/*在方向更改后防止 iOS 文本大小调整，而不禁用用户缩放*/
html {
    -ms-text-size-adjust: 100%;
    -webkit-text-size-adjust: 100%;
}
/*删除默认边距*/
body {margin: 0;}
/*::: :::*/
/*链接样式中，解决 Chrome 与其他浏览器之间的 outline 不一致问题*/
a:focus {outline: thin dotted;}
a:active, a:hover {outline: 0;}
/*在所有浏览器中解决不一致和可变的字体大小*/
small {font-size: 80%;}
/*在 IE 8/9 中的 a 元素中删除边框*/
img {border: 0;}
```

17.4 设计响应式样式

17.3 节介绍了网页基本样式的设计，当然限于篇幅，我们没有面面俱到，读者可以参考示例源代码，了解更详细的样式代码。下面重点介绍页面的响应式样式设计过程。

17.4.1 创建可伸缩图像

在默认情况下，图像显示的尺寸是 HTML 中指定的 width 和 height 属性值。如果不指定这些属性值，图像就会自动按照其原始尺寸显示。此外，可以通过 CSS 以像素为单位设置 width 和 height。显然，当屏幕宽度有限时，按原始尺寸显示图像就不一定合适了。使用可伸缩图像技术，就可以让图像在可用空间内缩放，但不会超过其本来的宽度。



为图像添加如下类样式：

```
.post-photo,.post-photo-full {
    max-width: 100%;
}
```



这样可伸缩图像可以根据包含它们的元素（本例为 body）的尺寸按比例缩放。

一定要使用 max-width: 100%而不是 width: 100%。虽然二者都能让图像在容器内缩放，但是 width: 100%会让图像尽可能地填充容器，如果容器的宽度比图像宽，图像就会放大到超过其本来尺寸，有可能会显得较为难看。

上面样式对已经为 Retina 显示屏扩大到双倍大小的图像也适用。当然，双倍分辨率的图像的文件大小也会大很多。

可以使用 background-size 属性对背景图像进行缩放（对 IE8 无效）。更多信息参见 www.css3.info/preview/background-size/。

提示：还可以使用 video, embed, object{max-width: 100%;}让 HTML5 视频及其他媒体变成可伸缩的（同样也不要在 HTML 中为它们指定 width 和 height）。

17.4.2　创建弹性布局网格

创建弹性布局需要使用百分数宽度，并将它们应用于页面里的主要区域。

```
width: percentage;
```

其中 percentage 表示希望元素在水平方向上占据容器空间的比例。一般不必设置“width: 100%;”，因为默认设置为“display:block;”的元素，如 p 以及其他很多元素，或手动设置为“display: block;”的元素在默认情况下会占据整个可用空间。

作为可选的一步，对包含整个页面内容的元素设置“max-width: value;”，其中 value 表示希望页面最多可增加到的最大宽度。通常，value 以像素为单位，不过也可以使用百分数、em 值或其他单位的值。

还可以对元素设置基于百分数的 margin 和 padding 值。在本例页面中，对这些属性使用的是 em 值，这是一种常见的做法。内边距和外边距的 em 值相对于元素的 font-size。例如，如果其字体大小等价于 14px，则“width:10em;”会将宽度设置为 140px。而基于百分数的值则是相对于包含元素的容器的。

对于设置了 body {font-size: 100%;}的页面，对 font-size、margin、padding 和 max-width 使用 em 值还有一个好处：如果用户更改了浏览器默认字体大小，那么页面也会跟着变大或变小。

注意：将 box-sizing 属性设置为 borderbox，就可以很方便地对拥有水平方向内边距（使用 em 或其他的单位）的元素定义宽度，而不必进行复杂的数学计算来找出百分数的值。这对响应式页面来说很方便。

17.4.3　实现媒体查询

可以使用两种方式针对特定的媒体类型定位 CSS。

第 1 种方式是使用 link 元素的 media 属性，位于 head 内。例如：

```
<head>
<link rel="stylesheet" href="your-styles.css" media="screen" />
</head>
```



第 2 种方式是在样式表中使用@media 规则。例如：

```
/*只用于打印的样式*/
@media print {
    header[role="banner"] nav, .ad {
        display: none;
    }
}
```



通过@media print 规则可以创建专门为打印浏览器里的页面定义的样式，也可以将它们与为其他媒体定义的样式放在一起。

此外，还有第 3 种方式，即使用@import 规则，不过一般不建议使用这种方法，因为它会影响性能。

媒体查询增强了媒体类型方法，允许根据特定的设备特性定位样式。要调整网站的呈现样式，让其适应不同的屏幕尺寸，采用媒体查询特别方便。

CSS3 使用@media 规则定义媒体查询，简化语法格式如下：

```
@media [only | not]? <media_type> [and <expression>]* | <expression> [and <expression>]* {
    /*CSS 样式列表*/
}
```

参数简单说明如下。

☑ <media_type>：指定媒体类型。

☑ <expression>：指定媒体特性。放在一对圆括号中，如(min-width:400px)。

☑ 逻辑运算符，如 and（逻辑与）、not（逻辑否）、only（兼容设备）等。

媒体特性包括 13 种，接受单个的逻辑表达式作为值，或者没有值。大部分特性接受 min-或 max-前缀，用来表示大于等于或者小于等于的逻辑，以此避免使用大于号（>）和小于号（<）字符。下面列出了可以包含在媒体查询里的媒体特性。

☑ width（宽度）。

☑ height（高度）。

☑ device-width（设备宽度）。

☑ device-height（设备高度）。

☑ orientation（方向）。

☑ aspect-ratio（高宽比）。

☑ device-aspect-ratio（设备高宽比）。

☑ color（颜色）。

☑ color-index（颜色数）。

☑ monochrome（单色）。

☑ resolution（分辨率）。

☑ scan（扫描）。

☑ grid（栅格）。

还有一些非标准的媒体特性：

☑ -webkit-device-pixel-rati（WebKit 设备像素比）。

☑ -moz-device-pixel-ratio（Mozilla 设备像素比）。

除了 orientation、scan 和 grid 以外，上述属性均可添加 min-和 max-前缀。min-前缀定位的是“大于等于”对应值的目标，而 max-前缀定位的则是“小于等于”对应值的目标。



CSS3 媒体查询规范中列出了关于所有媒体特性的描述（www.w3.org/TR/css3-mediaqueries/#media1）。以下是媒体查询的基本语法。

☑　指向外部样式表的链接：

```
//下面代码定义了如果页面通过屏幕呈现，且屏幕宽度不超过 480px，则加载 shetland.css 样式表
<link rel="stylesheet" type="text/css" media="screen and (max-device-width: 480px)" href="shetland.css" />
```



Note

☑　位于样式表中的媒体查询：

```
@media logic type and (feature: value) {
    /*目标 CSS 样式规则写在这里*/
}
```

对于响应式页面，大多数情况下需要将媒体查询放到样式表中。

```
/*常规样式写在这里。
每个设备都能获取它们，除非被媒体查询中的样式规则覆盖*/
body {font: 200%/1.3 sans-serif;}
p {color: green;}
/*以下针对不同的设备进行定制*/
@media (max-width: 600px) {
    /*匹配界面宽度小于等于 600px 的设备*/
}
@media (min-width: 400px) {
    /*匹配界面宽度大于等于 400px 的设备*/
}
@media (max-device-width: 800px) {
    /*匹配设备（不是界面）宽度小于等于 800px 的设备*/
}
@media (min-device-width: 600px) {
    /*匹配设备（不是界面）宽度大于等于 600px 的设备*/
}
```

本例设计的媒体查询样式如下：

```
/*480-767-only*/
@media only screen and (min-width: 30em) and (max-width: 47.9375em) {
    /*边栏*/
    .about {overflow: hidden;}
    .about img {
        float: left;
        margin-right: 15px;
    }
}
/*600-767-only*/
@media only screen and (min-width: 37.5em) and (max-width: 47.9375em) {
    /*边栏*/
    .about {padding-bottom: 1.4em;}
}
/*768+*/
@media only screen and (min-width: 48em) {
    h1 {font-size: 2.25em;}
```





Note

```
        /*报头*/
        .masthead {padding-top: 10px;}
        .nav-main {margin-bottom: 0;}
        /*主要内容*/
        .container {
            background: url(img/bg.png) repeat-y 65.938% 0;
            padding-bottom: 1.9375em;
        }
        main {
            float: left;
            width: 62.5%;
        }
        main > .post:first-child > h1 {margin-top: 0.904em;}
        .post-blurb p {
            font-size: 1em;
            line-height: 1.4;
        }
        .post-footer {
            padding-bottom: .7em;
            padding-top: .7em;
        }
        .footer p {font-size: 0.688em;}
        .pagination {margin-top: 45px;}
        /*边栏*/
        .sidebar {
            float: right;
            margin-top: 1.875em;
            width: 31.25%;
        }
        .about p {font-size: 0.813em;}
        /*页脚*/
        footer[role="contentinfo"] {border-top: 1px solid #cacbcb;}
    }
    /*---结束媒体查询----*/
```

最后，还需要在头部区域设置视口，即视觉区域。

```
<meta name="viewport" content="width=device-width, initial-scale=1" />
```

这段代码的重点是 width=device-width。有了这条语句，视觉区域的宽度会被设成与设备宽度（对 iPhone 来说为 320px）相同的值，因此在纵向模式下该宽度的页面内容会填充整个屏幕。如果不包含这一语句，使用媒体查询的 min-width 和 maxwidth 特性将不会得到预期的结果。

代码中的 initial-scale=1 部分对 width 和 device-width 值没有影响，但通常会包含这一语句。它将页面的默认缩放级别设成了 100%，换成纵向模式也一样。如果不设置 initial-scale=1，在 iPhone 中，手机从纵向模式改为横向模式时，网页会被放大，从而让布局与纵向模式一致。

提示：视觉区域（viewport）指的是浏览器（包括桌面浏览器和移动浏览器）显示页面的区域。它不包含浏览器地址栏、按钮等，只是浏览区域。

媒体查询的 width 特性对应的是视觉区域的宽度。不过，device-width 特性不是，它指的



是屏幕的宽度。在移动设备（如 iPhone）上，默认情况下这两个值通常不一样。Mobile Safari（iPhone 的浏览器）的视觉区域默认为 980px，但 iPhone 的屏幕只有 320px（iPhone 的 Retina 显示屏的屏幕分辨率有 640px，但它们是在相同的空间挤入两倍的像素，因此设备宽度仍为 320px）。

因此，iPhone 会像设为 980px 的桌面浏览器那样显示页面，并将页面缩小以适应 320px 屏幕宽度（在纵向模式下）。结果，当在 Mobile Safari 中浏览大部分为桌面浏览器建立的网站时，会显示将这些网站缩小了的样子。在横向模式下也是这样处理的，只不过宽度为 480px（iPhone5 是 568px）。如果不进行放大，页面通常是难以阅读的（注意不同设备的默认视觉区域宽度并不相同）。



Note

17.4.4　组合样式

理解了可伸缩图像、弹性布局和媒体查询的知识之后，就可以将它们组合在一起，创建响应式网页。本节将重点介绍页面扩张或收缩时切换内容的显示方式所需要考虑的要点。重要的是了解如何建立响应式网站，以及用于实现响应式网站的媒体查询类型。完整的样式需要读者参考示例源代码。注意，并不需要先做出一个定宽的设计，再将它转换成响应式的页面。

【操作步骤】

第 1 步，创建内容和 HTML。

在动手设计响应式设计之前，应该把内容和结构设计妥当。如果使用临时占位符设计和构建网站，当填入真正的内容以后，可能会发现形式与内容结合得不好。因此，应该尽可能地将内容采集工作提前。具体操作就不再展开。

在 head 元素中添加<meta name= "viewport" content="width=device-width, initialscale=1"/>。关于这行代码的作用，可以参考 17.4.3 节说明。

第 2 步，遵循移动优先为页面设计样式，推荐在设计网页时也遵循这一点。

首先为所有的设备提供基准样式。这同时也是旧版浏览器和功能比较简单的设备显示的内容。基准样式通常包括基本的文本样式（字体、颜色、大小）、内边距、边框、外边距和背景（视情况而定），以及可伸缩图像的样式。通常，在这个阶段，需要避免让元素浮动或对容器设定宽度，因为最小的屏幕并不够宽。内容将按照常规的文档流由上到下进行显示。

网站的目标是在单列显示样式中是清晰的、中看的。这样，网站对所有的设备（无论新旧）都具有可访问性。在不同设备下，外观可能有差异，不过这是在预期之内的，完全可以接受。

第 3 步，从基本样式开始，使用媒体查询逐渐为更大的屏幕或其他媒体特性定义样式，如 orientation。大多数时候，min-width 和 max-width 媒体查询特性是最主要的工具。

这是渐进增强在实战中的应用。处理能力较弱的（通常也是较旧的）设备和浏览器会根据它们能理解的 CSS 显示网站相对简单的版本。处理能力较强的设备和浏览器则显示增强的版本。所有人都能获取到网页的内容。

```
/*基准样式
----------------------------------*/
body {
    font: 100%/1.2 Georgia, "Times New Roman", serif;
    margin: 0;
    ...
```





Note

```
}
* { /*参见示例源代码*/
      -webkit-box-sizing: border-box;
      -moz-box-sizing: border-box;
      box-sizing: border-box;
}
.page {
      margin: 0 auto;
      max-width: 60em; /*960px*/
}
h1 {
      font-family: "Lato", sans-serif;
      font-size: 2.25em; /*36px/16px*/
      font-weight: 300;
      ...
}
.about h2, .mod h2 {font-size: .875em; /*15px/16px*/}
.logo, .social-sites,.nav-main li {text-align: center;}
/*创建可伸缩图像*/
.post-photo, .post-photo-full,.about img, .map {
      max-width: 100%;
}
```

第 4 步，应用于所有视觉区域（小屏幕和大屏幕设备）的基准样式示例效果如图 17.2 所示。这些样式规则与 17.4.3 节介绍的代码是类似的，只是它们没有由媒体查询包围。注意，本例为整个页面设定了 60em 的最大宽度（通常等价于 960px），并使用 auto 外边距让其居中，还让所有的元素使用“boxsizing:border-box;”，将大多数图像设置为可伸缩图像。





图 17.2　页面结构默认显示效果

图 17.2 是仅应用了基础样式的页面效果，这个页面在所有的浏览器中都是线性的，右侧的部分出现在左侧部分的下面；也是不支持媒体查询的旧浏览器中页面的显示效果。在这种状态下，依然保持了很高的可用性。由于没有设定容器宽度，因此在桌面浏览器中查看页面时，内容的宽度会延伸至



整个浏览器窗口的宽度。

第 5 步，逐步完善布局，使用媒体查询为页面中的每个断点（breakpoint）定义样式。断点即内容须作适当调整的宽度。在本例中，应用基准样式规则后，为下列断点创建了样式规则。



> 注意：对于每个最小宽度（没有对应的最大宽度），样式定位的是所有宽度大于该 min-width 值的设备，包括台式机及更早的设备。

第 6 步，最小宽度为 20em，通常为 320px。定位纵向模式下的 iPhone、iPod touch、各种 Android 以及其他移动电话。

```
/*基准样式
----------------------------------*/
...
/*20em（大于等于 320px）
----------------------------------*/
@media only screen and (min-width:20em) {
    .nav-main li {
        border-left: 1px solid #c8c8c8;
        display: inline-block;
        text-align: left;
    }
    .nav-main li:first-child {
        border-left: none;
    }
    .nav-main a {
        display: inline-block;
        font-size: 1em;
        padding: .5em .9em .5em 1.15em;
    }
}
```

这里针对视觉区域不小于 20em 宽的浏览器修改了主导航的样式。设计 body 元素字体大小为 16px 的情况下，20em 通常等价于 320px，因为 20×16=320。这样，链接会出现在单独的一行，而不是上下堆叠，如图 17.3 所示。



图 17.3　小屏显示效果

这里没有将这些放到基础样式表中，因为有的手机屏幕比较窄，可能会让链接显得很局促，或者分两行显示。

第 7 步，最小宽度为 30em，通常为 480px，如图 17.4 所示。定位大一些的移动电话，以及横向模式下的大量 320px 设备（iPhone、iPodtouch 及某些 Android 机型）。




Note



图 17.4　中屏显示效果

第 8 步，最小宽度介于 30em（通常为 480px）和 47.9375em（通常为 767px）之间。这适用于处于横向模式的手机、一些特定尺寸的平板电脑（如 Galaxy Tab 和 Kindle Fire），以及比通常情况更窄的桌面浏览器。

第 9 步，最小宽度为 48em，通常为 768px。这适应于常见宽度及更宽的 iPad、其他平板电脑和台式机的浏览器。

主导航显示为一行，每个链接之间由灰色的竖线分隔。这个样式会在 iPhone（以及很多其他手机）中生效，因为它们在纵向模式下是 320px 宽。如果希望报头更窄一些，可以让标识居左，社交图标居右。将这种样式用在下一个媒体查询中，代码如下：

```
/*基准样式*/
...
/*20em（大于等于 320px）*/
@media only screen and (min-width: 20em) {
    ...
}
/*30em（大于等于 480px）*/
@media only screen and (min-width: 30em) {
    .masthead {position: relative;}
    .social-sites {
        position: absolute;
        right: -3px;
        top: 41px;
    }
    .logo {
        margin-bottom: 8px;
        text-align: left;
    }
    .nav-main {margin-top: 0;}
}
```

现在，样式表中有了定位视觉区域至少为 30em（通常为 480px）的设备的媒体查询。这样的设备包括屏幕更大的手机，以及横向模式下的 iPhone。这些样式会再次调整报头。

第 10 步，在更大的视觉区域，报头宽度会自动调整。

```
/*30em（大于等于 480px）*/
@media only screen and (min-width: 30em) {
    ... 报头样式 ...
    .post-photo {
        float: left;
```



```
        margin-bottom: 2px;
        margin-right: 22px;
        max-width: 61.667%;
    }
    .post-footer {clear: left;}
}
```



第 11 步，继续在同一个媒体查询块内添加样式，让图像向左浮动，并减少其 max-width，从而让更多的文字可以浮动到其右侧。文本环绕在浮动图像周围的断点可能跟此处用的不同。这取决于哪些断点适合内容和设计。为适应更宽的视觉区域，一般不会创建超过 48em 的断点。也不一定要严格按照设备视觉区域的宽度创建断点。如果一个基于 min-width:36em 的断点非常适合内容，就可以大胆地使用这个断点。

```
/*基准样式*/
/*20em（大于等于 320px）*/
@media only screen and (min-width: 20em) {
    ...
}
/*30em（大于等于 480px）*/
@media only screen and (min-width: 30em) {
    ...
}
/*30em – 47.9375em（在 480px 和 767px 之间）*/
@media only screen and (min-width: 30em) and (max-width: 47.9375em) {
    .about { /*self-clear float*/
        overflow: hidden;
    }
    .about img {
        float: left;
        margin-right: 15px;
    }
}
```

第 12 步，让“关于自己”图像向左浮动。不过，这种样式仅当视觉区域的宽度在 30em 和 47.9375em 之间时才生效，超过这个宽度会让布局变成两列布局，“关于自己”文字会再次出现在图像的下面。浮动的“关于自己”图像已显示为其本来的尺寸（270px 宽），它旁边的空间太小，无法很好地容纳文本。这就是之前减少其 max-width 的原因。

```
/*基准样式*/
...
/*20em（大于等于 320px）*/
@media only screen and (min-width: 20em) {
    ...
}
/*30em（大于等于 480px）*/
@media only screen and (min-width: 30em) {
    ...
}
/*30em～47.9375em（在 480px 和 767px 之间）*/
@media only screen and (min-width: 30em) and  (max-width: 47.9375em) {
```





Note

```
        ...
}
/*48em（大于等于 768px）*/
@media only screen and (min-width: 48em) {
    .container {
        background: url(../img/bg.png) repeat-y 65.9375% 0;
        padding-bottom: 1.875em;
    }
    main {
        float: left;
        width: 62.5%; /*600px/960px*/
    }
    .sidebar {
        float: right;
        margin-top: 1.875em;
        width: 31.25%; /*300px/960px*/
    }
    .nav-main {margin-bottom: 0;}
}
```

这是最终的媒体查询，定位至少有 48em 宽的视觉区域，如图 17.5 所示。该媒体查询对大多数桌面浏览器来说都为真，除非用户让窗口变窄。它同时也适用于纵向模式下的 iPad 及其他一些平板电脑。



图 17.5　大屏显示效果

在桌面浏览器（尽管要宽一些）中也是类似的。由于宽度是用百分数定义的，因此主体内容栏和附注栏会自动伸展。

第 13 步，在发布响应式页面之前，应在移动设备和桌面浏览器上对其测试一遍。构建响应式页面时，用户可以放大或缩小桌面浏览器的窗口，模拟不同手机和平板电脑的视觉区域尺寸，然后对样式进行相应的调整。这当然是一种不够精细的办法，但它确实有助于建立有效的样式，从而减少在真实设备上优化网站的时间。

第 14 步，对 Retina 及类似显示屏使用媒体查询。针对高像素密度设备，可以使用下面的媒体查询：

```
@media (-o-min-device-pixel-ratio: 5/4),(-webkit-min-device-pixel-ratio:1.25),(min-resolution: 120dpi) {
    .your-class {
```



```
            background-image:url(sprite-2x.png);
            background-size: 200px 150px;
        }
    }
```



注意：background-size 设置成了原始尺寸，而不是 400px×300px。这样会让图像缩小，为原始尺寸创建的样式对 2x 版本也有效。

17.4.5　兼容旧版 IE

对于移动优先的方法，有一点需要注意，就是 IE8 及以下的版本不支持媒体查询。这意味着这些浏览器只会呈现媒体查询以外的样式，即基准样式。目前，使用 IE6 和 IE7 的用户已经非常少了。因此，真正需要考虑的是 IE8，它在全世界所占的份额不到 9%，且这个数字还在下降（详情参见 http://gs.statcounter.com）。

对于 IE8（及更早的版本），有 3 种解决方法。

☑ 什么都不做，让网站显示基本的版本。
☑ 为它们单独创建一个样式表，让它们显示网站最宽的版本（不会形成响应式的网页）。一种做法是复制一份常规的样式表，将其命名为 old-ie.css 之类的文件名。将媒体查询语句去掉，但保留其中的样式规则。在 HTML 中添加条件注释，从而让不同的浏览器都能找到正确的样式表。
☑ 如果希望页面有响应式的效果，就在页面中引入 respond.min.js。Scott Jehl 创建了这段简短的代码，它让 min-width 和 max-width 媒体查询对旧版 IE 也有效。

```
<!--[if lt IE 9]>
    <script src="js/respond.min.js"></script>
<![endif]-->
```

设置好以后，IE8 及以下版本会理解 min-width 和 max-width 媒体查询，并呈现相应的样式。这样做就没有必要将 IE 样式表分离出来。script 元素外围的条件注释是可选的，不过如果包含，就只有 IE8 及以下版本会加载 respond.min.js，它让 IE8 用户也能看到网站的完整布局。

提示：可以访问 https://github.com/scottjehl/Respond，下载 Respond.js。下载到计算机后，打开该 ZIP 文件，然后将 respond.min.js 复制到网站中。



案例开发：酒店预订微信 wap 网站

本章将介绍如何设计一个预订酒店的手机应用网站。该网站以 Bootstrap 框架为技术基础，页面设计简洁、明亮，功能以“微”为核心，为浏览者提供一个迷你、简单、时尚的设计风格，与 Bootstrap 框架风格完美融和，非常适合移动应用和推广。

【学习重点】

⏭ 设计符合移动设备使用的页面。

⏭ 能够根据 Bootstrap 框架自定义样式。

⏭ 掌握扁平化设计风格的基本方法。



18.1　设 计 思 路



Note



视频讲解

与第 17 章案例相比，本章案例相对复杂，在开始设计之前，先来理清一下设计思路，下面仅做简单介绍。

18.1.1　内容

网站涉及的内容可能很多，单从网页设计的角度看，内容主要包括图片和文字。本例素材具体存放文件夹说明如下。

☑ Images：图片等多媒体素材。
☑ styles：样式表文件。
☑ Scripts：JavaScript 脚本文件。
☑ Pictures：宣传的图片。
☑ Member：后台支持文件，本章暂不介绍。
☑ help：帮助文件。
☑ dialog：jQuery 插件文件，模态对话框插件。
☑ calendar：日历插件。

本例所需要的素材不是很多，但是涉及的文件比较多。

18.1.2　结构

视频讲解

本例主要包含下面几个文件，简单说明如下。

☑ index.html：首页。
☑ Activitys.html：最新活动页面。
☑ CityList.html：城市列表页面。
☑ Gift.aspx.html：礼品页模板，供后台参考使用。
☑ GiftList.html：礼品商城。
☑ Hotel.aspx.html：预订酒店，选择房型模板，供后台参考使用。
☑ Hotel.aspxcheckInDate.html：房型和日期选择页面。
☑ HotelInfo.aspx.html：酒店信息介绍模板，供后台参考使用。
☑ HotelList.aspxcheckInDate.html：所选城市的相关酒店信息列表页面。
☑ HotelReview.aspx.html：用户评价页面。
☑ login.html：用户登录页面。
☑ News.aspx.html：酒店新闻页面。

结构不仅仅包含文件，更多涉及页面内容，根据内容搭建页面结构，在下面各节中会逐一介绍每个页面的结构框架。

18.1.3　效果



视频讲解

下面使用 Opera Mobile Emulator 来预览一下网站整体效果，以便在分页设计时有一个整体把握。







首先，打开 index.html 页面，显示效果如图 18.1 所示。

首页以扁平化进行设计，包含 6 个导航图标色块，单击第一个图标色块“预订酒店”，进入“选择城市”页面，在该页面选择要入住的酒店，页面效果如图 18.2 所示。

在首页单击“最新活动”图标色块，进入“最新活动”页面，在该页面显示酒店促销活动的相关信息，如图 18.3 所示。





图 18.1 首页页面设计效果



图 18.2 选择要入住的酒店效果



图 18.3 “最新活动”页面效果

在首页单击“我的订单”图标色块，可以查看个人订单信息，如果没有登录，则显示登录表单，如图 18.4 所示。

在首页单击“我的格子”图标色块，进入“个人信息中心”页面，如果没有登录，则显示登录表单。在首页单击“礼品商城”图标色块，进入“礼品商城”页面，如图 18.5 所示。

在首页单击“帮助咨询”图标色块，将进入“帮助咨询”页面，可咨询相关帮助信息，如图 18.6 所示。



图 18.4 登录表单页面



图 18.5 “礼品商城”页面效果



图 18.6 “帮助咨询”页面效果







18.2　设计首页

首页是一个简单的导航列表。

【操作步骤】

第 1 步，打开 index.html 文件，首先在头部区域导入框架文件。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0;">
<meta content="yes" name="apple-mobile-web-app-capable">
<link href="styles/bootstrap.min.css" rel="stylesheet">
<link href="styles/bootstrap-responsive.css" rel="stylesheet">
<link href="styles/NewGlobal.css" rel="stylesheet">
<script src="Scripts/jquery-1.7.2.min.js"></script>
<script src="Scripts/bootstrap.min.js"></script>
</head>
<body>
</body>
</html>
```

第 2 步，设计导航列表结构，使用<div class="container">布局容器。

```
<div class="container">
    <div class="header"> <img src="Images/logo.png" style="height: 40px; margin: 10px 0px 0px 15px"> </div>
    <div style="padding:0 5px 0 0;">
</div>
```

其包含两部分：

第 1 行为标题栏，显示网站 Logo，本例以一张大图代替显示，效果如图 18.7 所示。



图 18.7　设计首页标题栏效果

大图为 PNG 格式，镂空白色文字，然后使用 CSS 设计标题栏背景色为绿色。

```
.header {
    background:#6ac134;
    height: 60px;
    position: relative;
    width: 100%;
}
```

第 2 行为导航列表结构，使用 3 个<ul class="unstyled defaultlist pt20">标签堆叠显示，每个列表框包含两个列表项目，水平布局，效果如图 18.8 所示。



Note

```
<div style="padding:0 5px 0 0;">
    <ul class="unstyled defaultlist pt20">
        <li class="f"> <a href="CityList.html">
            <h3>预订酒店</h3>
            <figure class="jp_icon"></figure>
            </a> </li>
        <li class="h"> <a href="Activitys.html">
            <h3>最新活动</h3>
            <figure class="jd_icon"></figure>
            </a> </li>
    </ul>
    <ul class="unstyled defaultlist">… </ul>
    <ul class="unstyled defaultlist">…</ul>
</div>
```

每个导航图标使用<a>标签包裹，里面包含文字和字体图标，然后在<li>列表项目上面定义不同皮肤颜色，<li>标签浮动显示，实现一行两列排版布局。

第 3 步，在页面底部插入<div class="footer">包含框，定义网站版权信息区域，结构如下，效果如图 18.9 所示。

```
<div class="footer">
    <div class="gezifooter"> <a href="#" class="ui-link">酒店预订</a> <font color="#878787">|</font> <a href=
"#" class="ui-link">我的订单</a> <font color="#878787">|</font> <a href="#" class="ui-link">我的格子</a> </div>
    <div class="gezifooter">
        <p style="color:#bbb;">格子微酒店连锁 &copy; 版权所有 2012-2017</p>
    </div>
</div>
```



图 18.8 设计首页导航图标效果

酒店预订 | 我的订单 | 我的格子
格子微酒店连锁 © 版权所有 2012-2017

图 18.9 设计页脚信息显示效果



视频讲解

18.3 设计登录页

打开 login.html 页面，该页面包含 3 部分：顶部标题栏、底部脚注栏和中间的登录表单结构。

标题栏采用标准的移动设备布局样式，左右为导航图标，中间为标题文字，效果如图 18.10 所示。

```
<div class="header"> <a href="index.html" class="home"> <span class="header-icon header-icon-home"></span>
<span class="header-name">主页</span> </a>
    <div class="title" id="titleString">登录</div>
```



```
        <a href="javascript:history.go(-1);" class="back"> <span class="header-icon header-icon-return"></span>
<span class="header-name">返回</span> </a>
    </div>
```



图 18.10　标题栏设计效果



标题栏图标以文字图标形式设计，这样方便与文字定义相同的颜色，为<div class="header">设计绿色背景，营造一种扁平设计风格。

中间位置显示表单结构，代码如下：

```
    <div class="container width80 pt20">
        <form name="aspnetForm" method="post" action="login.aspx?ReturnUrl=%2fMember%2fDefault.aspx"
id="aspnetForm" class="form-horizontal">
            <div>
                <input type="hidden" name="__EVENTTARGET" id="__EVENTTARGET" value="">
                <input type="hidden" name="__EVENTARGUMENT" id="__EVENTARGUMENT" value="">
                <input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE" value="1">
            </div>
            <div>
                <input type="hidden" name="__EVENTVALIDATION" id="__EVENTVALIDATION" value=
"/wEWBQLZmqilDgLJ4fq4BwL90KKTCAKqkJ77CQKI+JrmBdPJophKZ3je4aKMtEkXL+P8oASc">
            </div>
            <div class="control-group">
                <input name="ctl00$ContentPlaceHolder1$txtUserName" type="text" id="ctl00_ContentPlaceHolder1_
txtUserName" class="input width100 " style="background: none repeat scroll 0 0 #F9F9F9;padding: 8px 0px 8px 4px"
placeholder="请输入手机号/身份证/会员卡号">
            </div>
            <div class="control-group">
                <input name="ctl00$ContentPlaceHolder1$txtPassword" type="password" id="ctl00_ContentPlaceHolder1_
txtPassword" class="width100 input" style="background: none repeat scroll 0 0 #F9F9F9;padding: 8px 0px 8px 4px"
placeholder="默认密码为证件号后 4 位">
            </div>
            <div class="control-group">
                <label class="checkbox fl">
                    <input name="ctl00$ContentPlaceHolder1$cbSaveCookie" type="checkbox" id="ctl00_
ContentPlaceHolder1_cbSaveCookie" style="float: none;margin-left: 0px;">
                    记住账号 </label>
                <a class="fr" href="GetPassword.aspx">忘记密码？ </a> </div>
            <div class="control-group"> <span class="red"></span> </div>
            <div class="control-group">
                <button onclick="__doPostBack('ctl00$ContentPlaceHolder1$btnOK','')" id="ctl00_ContentPlaceHolder1_
btnOK" class="btn-large green button width100">立即登录</button>
            </div>
            <div class="control-group"> 还没账号？<a href="Reg.aspx@ReturnUrl=_252fMember_252fDefault.aspx"
id="ctl00_ContentPlaceHolder1_RegBtn">立即免费注册</a> </div>
            <div class="control-group"> 或者使用合作账号一键登录：<br>
                <a class="servIco ico_qq" href="qlogin.aspx"></a> <a class="servIco ico_sina" href="default.htm">
</a> </div>
```



```
        </form>
    </div>
```

在表单中通过<input type="hidden">隐藏控件，传递用户附加信息，借助 Bootstrap 表单控件美化效果，如图 18.11 所示。提交按钮使用 Bootstrap 的风格设计块状显示，在整个页面中显得很大气、可触。



Note

```
    <button class="btn-large green button width100">立即登录</button>
```



图 18.11　登录表单设计效果



视频讲解

18.4　选择城市

打开 CityList.html 页面，该页面提供一个交互界面，供用户选择要入住酒店所在的城市。

该页面标题栏和脚注栏与其他页面设计相同，在此不再重复，下面主要设计交互表单界面。表单结构如下：

```
    <div class="container width90 pt20">
        <form class="form-horizontal" action="HotelList.aspx" method="get" id="form1">
            <ul class="search-group unstyled">
                <li>
                    <div class="coupon-nav coupon-nav-style"> <span class="search-icon location-icon"></span>
<span class="coupon-label">选择城市：</span> <span class="coupon-input"> <span style="font-size: 16px; line-height:
35px;" id="cityname">全部城市</span></span> </div>
                    <div class="citybox"> <span cityid="0">全部</span> <span cityid="771">南宁</span> <span
cityid="773">桂林</span> <span cityid="371">郑州</span> </div>
                </li>
                <li>
                    <div class="coupon-nav coupon-nav-style"> <span class="search-icon time-icon"></span>
<span class="coupon-label">入住日期：</span> <span class="coupon-input"><a id="datestart" class="datebox"
href="javascript:void(0)"><span class="ui-icon-down"></span></a></span> </div>
                    <div id="dp_start" class="none">
                        <div id="datepicker_start"></div>
                    </div>
                </li>
                <li>
```



```
                <div class="coupon-nav coupon-nav-style"> <span class="search-icon time-icon"></span> <span
class="coupon-label">离店日期：</span> <span class="coupon-input"><a id="dateend"  class="datebox"  href=
"javascript:void(0)"><span class="ui-icon-down"></span></a></span> </div>
                <div id="dp_end" class="none">
                    <div id="datepicker_end"></div>
                </div>
            </li>
        </ul>
        <input id="checkInDate" name="checkInDate" value="2017-04-11" type="hidden">
        <input id="checkOutDate" name="checkOutDate" value="2017-04-12" type="hidden">
        <input id="cityID" name="cityID" value="0" type="hidden">        <div class="control-group tc">
            <button class="btn-large green button width80" style="padding-left:0px;padding-right: 0px;" ID=
"btnOK" >
            <A href="HotelList.aspxcheckInDate.html">立即查找</A>
            </button>
        </div>
        <div class="control-group tc"> <a href="NearHotel.aspx" style="padding-left:0px;padding-right: 0px;"
class="btn-large green button width80">附近酒店</a> </div>
    </form>
</div>
```

为了方便 JavaScript 脚本控制，整个页面没有使用传统的表单控件来设计，而是通过 JavaScript+CSS 来设计，界面效果如图 18.12 所示。

点击“选择城市”选项，将会滑出城市列表面板，如图 18.13 所示，用户可以选择目标城市。



图 18.12　查找酒店界面



图 18.13　选择城市

在城市列表面板中选择一个城市，然后在下面选项中选择入住日期，效果如图 18.14 所示。



图 18.14　选择日期



用户选择的日期通过 JavaScript 显示在界面中，同时赋值给隐藏控件，以便传递给服务器进行处理。交互控制的 JavaScript 代码如下：



Note

```
<script type="text/javascript">
    (function ($, undefined) {
        $(function () {//dom ready
            var open = null, today = new Date();
            var beginday = '2017-04-11';
            var endday = '2017-04-12';
            //设置开始时间为今天
            $('#datestart').html(beginday + '<span class="ui-icon-down"></span>');
            //设置结束时间
            $('#dateend').html(endday +
                '<span class="ui-icon-down"></span>');
            $('#datepicker_start').calendar({//初始化开始时间的 datepicker
                date: $('#datestart').text(), //设置初始日期为文本内容
                //设置最小日期为当月第一天，既上一月的不能选
                minDate: new Date(today.getFullYear(), today.getMonth(), today.getDate()),
                //设置最大日期为结束日期，结束日期以后的天不能选
                maxDate: new Date(today.getFullYear(), today.getMonth(), today.getDate() + 25),
                select: function (e, date, dateStr) {//当选中某个日期时
                    var day1 = new Date(date.getFullYear(), date.getMonth(), date.getDate() + 1);
                    //将结束时间的 datepicker 的最小日期设成所选日期
                    $('#datepicker_end').calendar('minDate', day1).calendar('refresh');
                    $('#dp_start').toggle();
                    //把所选日期赋值给文本
                    $('#datestart').html(dateStr + '<span class="ui-icon-down"></span>').removeClass('ui-state-
active');
                    $('#checkInDate').val(dateStr);
                    $('#dateend').html($.calendar.formatDate(day1) + '<span class="ui-icon-down"></span>').
removeClass('ui-state-active');
                    $('#checkOutDate').val($.calendar.formatDate(day1));
                }
            });
            $('#datepicker_end').calendar({                    //初始化结束时间的 datepicker
                date: $('#dateend').text(),                    //设置初始日期为文本内容
                minDate: new Date(today.getFullYear(), today.getMonth(), today.getDate() + 1),
                maxDate: new Date(today.getFullYear(), today.getMonth(), today.getDate() + 16),
                select: function (e, date, dateStr) {          //当选中某个日期时
                    //收起 datepicker
                    open = null;
                    $('#dp_end').toggle();
                    //把所选日期赋值给文本
                    $('#dateend').html(dateStr + '<span class="ui-icon-down"></span>').removeClass('ui-state-
active');
                    $('#checkOutDate').val(dateStr);
                }
            });
            $('#datestart').click(function (e) {              //展开或收起日期
                $('#datestart').removeClass('ui-state-active');
```



```
                var type = $(this).addClass('ui-state-active').is('#datestart') ? 'start': 'end';
                $('#dp_start').toggle();
            }).highlight('ui-state-hover');
            $('#cityname').click(function (e) {
                $('.citybox').toggle();
            });
            $('.citybox span').click(function (e) {
                $('#cityname').text($(this).text());
                $('.citybox').toggle();
                $('#cityID').val($(this).attr("cityId"));
            });
            $('#dateend').click(function (e) {                    //展开或收起日期
                $('#dateend').removeClass('ui-state-active');
                var type = $(this).addClass('ui-state-active').is('#dateend') ? 'start': 'end';
                $('#dp_end').toggle();
            }).highlight('ui-state-hover');
        });
    })(Zepto);
</script>
```

18.5　选 择 酒 店


视频讲解

当用户在“选择城市”页面提交表单之后，将会跳转到 HotelList.aspx 页面，该页面为后台服务器处理文件，该文件将动态显示所在城市相关酒店信息列表，本例模拟效果如图 18.15 所示。



图 18.15　所选城市的酒店列表

页面基本结构如下（HotelList.aspxcheckInDate.html）：

```
<div class="container hotellistbg">
    <ul class="unstyled hotellist">
        <li> <a href="Hotel.aspxcheckInDate.html"> <img class="hotelimg fl" src="Pictures/1/5.jpg">
```





Note

```
                <div class="inline">
                    <h3>南宁秀灵店</h3>
                    <p>地址：秀灵路 55 号（出入境管理局旁）</p>
                    <p>评分：4.6 （1200 人已评）</p>
                </div>
                <div class="clear"></div>
                </a>
                <ul class="unstyled">
                    <li><a href="Hotel.aspx@id=5" class="order">预订</a></li>
                    <li><a href="Hotelmap.aspx@id=5" class="gps">导航</a></li>
                    <li><a href="Hotelinfo.aspx@id=5" class="reality">实景</a></li>
                </ul>
            </li>
            ......
        </ul>
    </div>
```

在该页面中可以选择特定酒店，并通过每个酒店底部的 3 个按钮预订酒店、查看酒店信息，或者进行导航等。



视频讲解

18.6　预订酒店

当用户在酒店列表页面选择一个酒店之后，将会跳转到 Hotel.aspx 页面，该页面为后台服务器处理文件，该文件将动态显示用户可选择的房型信息。本例模拟效果如图 18.16 所示。



图 18.16　选择房型

页面基本结构如下（Hotel.aspx.html）：

```
    <div class="container">
        <ul class="unstyled hotel-bar">
            <li class="first"> <a href="#BookRoom"  class="active">房型</a> </li>
            <li><a href="HotelInfo.aspx.html">简介</a></li>
```



```
                <li><a href="#">地图</a></li>
                <li><a href="Hotelreview.aspx.html">评论</a></li>
            </ul>
            <div id="BookRoom" class="tab-pane active fade in">
                <div class="detail-address-bar"> <img alt="" src="images/location_icon.png">
                    <p>秀灵路 55 号（出入境管理局旁）</p>
                </div>
                <div id="datetab" class="detail-time-bar"> <img alt="" src="images/calendar.png">
                    <p>04 月 11 日 - 04 月 12 日</p>
                    <span class="icon-down"></span> </div>
                <form action="hotel.aspx" method="get">
                    <div id="datebox" class="section none">
                        <div class="filter clearfix">
                            <p style="margin-bottom: 10px;display: block;">入住：<a id="datestart" href="javascript:
void(0)"><span class="ui-icon-down"></span></a></p>
                            <br>
                            <p> 离开： <a id="dateend" href="javascript:void(0)"><span class="ui-icon-down">
</span></a></p>
                        </div>
                        <div id="datepicker_wrap">
                            <div id="dp_start">
                                <p>入住时间：</p>
                                <div id="datepicker_start"></div>
                            </div>
                            <div id="dp_end">
                                <p>离开时间：</p>
                                <div id="datepicker_end"></div>
                            </div>
                        </div>
                        <div class="result">
                            <input type="submit" class="btn" value="确定修改">
                            <span class="btn" id="datecancel">取消</span> </div>
                        <input id="id" name="id" type="hidden" value="5">
                        <input id="CheckInDate" name="CheckInDate" type="hidden" value="2017-4-11">
                        <input id="CheckOutDate" name="CheckOutDate" type="hidden" value="2017-4-12">
                    </div>
                </form>
                  <ul class="unstyled roomlist">
                    <li>
                        <div class="roomtitle">
                            <div class="roomname">上下铺</div>
                            <div class="fr"> <em class="orange roomprice"> ¥134 起 </em> <a href='login.aspx@page=
_2Forderhotel.aspx&hotelid=5&roomtype=5&checkInDate=2017-4-11&checkOutDate=2017-4-12' title='立即预订'
class='btn btn-success iframe'>预订</a> </div>
                        </div>
                        <a class="fl roompic" bigsrc="Pictures/20130411152105m.jpg"> <img title="秀灵上下铺"
                                    src="Pictures/20130411152105s.jpg"></a> </li>
                    ...
                </ul>
                <div style="transform-origin: 0px 0px 0px; opacity: 1; transform: scale(1, 1);" class="hotel-prompt">
<span class="hotel-prompt-title" id="digxx">特别提示</span>
```



```
            <p>最早入住时间为中午 12:00，如需提前入住请联系客服。</p>
        </div>
    </div>
</div>
```



Note

在该页面顶部显示一行次级导航面板，分别为房型、简介、地图和评论。当点击“简介”选项，将会打开 HotelInfo.aspx 页面，该页面将会动态显示对应酒店的详细介绍。本例模板页面效果如图 18.17 所示。



图 18.17　查看酒店信息

页面结构如下（HotelInfo.aspx.html）：

```
<div class="container">
    <ul class="unstyled hotel-bar">
        <li class="first"> <a href="Hotel.aspx.html">房型</a> </li>
        <li><a href="HotelInfo.aspx"    class="active">简介</a></li>
        <li><a href="#">地图</a></li>
        <li><a href="HotelReview.aspx.HTML">评论</a></li>
    </ul>
    <div class="hotel-prompt "> <span class="hotel-prompt-title">酒店图片</span>
        <div id="slider" style="margin-top: 10px;">
            <div> <img src="Pictures/20121231113309m.jpg">
                <p>酒店外观</p>
            </div>
            <div> <img src="Pictures/20121231113406m.jpg">
                <p>大堂</p>
            </div>
            <div> <img src="Pictures/20121231113520m.jpg">
                <p>阳光大床房</p>
            </div>
        </div>
    </div>
    <div id="hotelinfo" class="hotel-prompt "> <span class="hotel-prompt-title">酒店简介</span>
        <p>格子微酒店南宁秀灵路店位于广西最著名大学广西大学东门旁，紧邻邕江边，周边超市、餐饮、
银行等配套设施完善，出行便利。 酒店倡导低碳环保，客房内配有 24 小时热水、wifi 网络、电视等设施，客房
```



```
虽小，设施齐全。酒店服务周到细致，是您出行的不错选择。 酒店开业时间 2012 年 12 月。</p>
            <p>地址：秀灵路 55 号（出入境管理局旁）</p>
            <p>电话：0771-3391588</p>
        </div>
    </div>
```



Note

如果在页面顶部点击“评论”选项，可以打开评论页面 HotelReview.aspx，了解网友对该酒店的评价信息列表，效果如图 18.18 所示。



图 18.18　查看用户评价信息

打开本例 HotelReview.aspx.html 模板页面，该页面的基本结构如下：

```
    <div class="container">
        <ul class="unstyled hotel-bar">
            <li class="first"> <a href="Hotel.aspx.HTML">房型</a> </li>
            <li><a href="HotelInfo.aspx.html">简介</a></li>
            <li><a href="#">地图</a></li>
            <li><a href="HotelReview.aspx.html" class="active">评论</a></li>
        </ul>
        <div class="hotel-comment-list">
            <div class="hotel-user-comment"> <span class="hotel-user"><img width="32" height="32" src="Pictures/2/
user01.png">会员李*清:</span>
                <div class="hotel-user-comment-cotent">
                    <p> 这次去这个房间有点烟味，住了这么多次只有这个有烟味~除了烟味都是一如既往的
好！ </p>
                    <span>2017-04-11</span> </div>
            </div>
            ...
        </div>
    </div>
```

上面重点介绍了酒店预订的完整流程，从选择城市，到选择酒店，再到选择房型、查看酒店信息和用户评论等，本例网站还包含其他辅助页面，这些页面设计风格相近，结构大致相同，这里不再详细展开。